Implicit Abstraction Heuristics for Cost-Optimal Planning

Michael Katz

Implicit Abstraction Heuristics for Cost-Optimal Planning

Research Thesis

In Partial Fulfillment of the
Requirements for the
Degree of Doctor of Philosophy

Michael Katz
Submitted to the Senate of
the Technion - Israel Institute of Technology

Av, 5770 Haifa August 2010

The Research Thesis Was Done Under The Supervision of Prof. Carmel Domshlak in the Faculty of Industrial Engineering and Management.

First of all, I would like to express my sincere gratitude to Prof. Carmel Domshlak, who has been my supervisor since the beginning of my study. He was not only the best supervisor one could possibly dream of, but also a great personal friend. I would also like to thank Erez Karpas for uncountable helpful discussions.

The Generous Financial Help Of Technion Is Gratefully Acknowledged.

Last, but not least, I would like to thank my family and friends, who supported me over the years. I could not do it without you.

Contents

List of Figures

List of Tables

Abstract

State-space search with explicit abstraction heuristics is at the state of the art of cost-optimal planning. These heuristics are inherently limited nonetheless, because the size of the abstract space must be bounded by some, even if very large, constant. Targeting this shortcoming, we introduce the notion of *(additive) implicit abstractions*, in which the planning task is abstracted by instances of tractable fragments of cost-optimal planning. We then introduce a concrete setting for this framework, called *fork-decomposition*, that is based on two novel fragments of tractable cost-optimal planning. The induced admissible heuristics are then studied formally and empirically. While our empirical evaluation demonstrates the accuracy of the fork decomposition heuristics, the runtime complexity of computing them poses an obvious tradeoff. Indeed, some of the power of the explicit abstraction heuristics comes from precomputing the heuristic function offline and then determining $h(s)$ for each evaluated state s by a very fast lookup in a "database." But implicit abstraction heuristics are, at first glance, a different story: while their calculation time is polynomial, it is far from being fast. To address this problem, we show that the time-per-node complexity bottleneck of the fork-decomposition heuristics can be successfully overcome. We demonstrate that an equivalent of the explicit abstraction notion of a "database" exists for the fork-decomposition abstractions as well, despite their exponential-size abstract spaces. We then verify empirically that heuristic search with the "databased" fork-decomposition heuristics favorably competes with the state of the art of cost-optimal planning.

Of course, as planning is known to be NP-hard even for extremely conservative planning formalisms, no heuristic should be expected to work well in all planning tasks. Thus, addditive ensembles of admissible heuristics are used in cost-optimal planning to exploit the individual strengths of numerous admissible heuristics. The same set of heuristics can, however, be composed in infinitely many ways, with the choice of composition directly determining the quality of the resulting heuristic estimate. Continuing our focus on abstraction heuristics, we describe a procedure that takes a deterministic planning problem, a forward-search state, and a set of admissible heuristics, and derives an *optimal* additive composition of these heuristics with respect to the given state. Most importantly, we show that this procedure is *polynomial-time* for arbitrary sets of all abstraction heuristics with which we are acquainted, including explicit abstractions such as pattern databases (regular or constrained) or merge-and-shrink, and implicit abstractions such as fork-decomposition or abstractions based on tractable constraint optimization over tree-shaped constraint networks.

Chapter 1

Introduction

AI problem solving is facing an inherent computational paradox. Most general AI reasoning tasks are known to be very hard, so much so that membership in NP is in itself sometimes perceived as "good news." If, however, the intelligence is somehow modeled by a computation, and the computation is delegated to the computers, then artificial intelligence has to escape the traps of intractability as much as possible. Planning is one such reasoning task, corresponding to finding a sequence of state-transforming actions that achieve a goal from the given initial state. It is well known that planning is intractable in general (Chapman, 1987), and that even the "simple" classical planning with propositional state variables is PSPACE-complete (Bylander, 1994).

While the planning community's interest in the formal complexity analysis of planning tasks has had its ups and downs, it is now understood that computational tractability is fundamental to all problem solving, for two practical reasons:

1. Planning tasks in automatically controlled real-world systems are believed to be highly structured. If discovered and exploited, this structure may allow for efficient planning (Klein, Jonsson, & Bäckström, 1998). But if this structure is ignored, a general-purpose planner is likely to go on tour in an exponential search space even for tractable tasks. Furthermore, when the overall system is required to provide some guarantees about its run-time complexity, system control cannot be based on worst-case intractable planning theory. The system should thus be designed so that planning for it will be provably tractable (Williams & Nayak, 1996, 1997).

2. Computational tractability can be an invaluable tool even for problems that fall outside all the known tractable fragments of planning. For instance, tractable fragments of planning provide the foundations for most (if not all) rigorous heuristic estimates employed in planning as heuristic search (Bonet & Geffner, 2001; Hoffmann, 2003; Helmert, 2006). This is in particular true for admissible heuristic functions for planning that are typically defined as the optimal cost of achieving the goals in an over-approximation of the planning task at hand. Such an approximation is obtained by relaxing or reformulating certain constraints in the specification of the original task, and the purpose of the approximation is to provide us with a provably tractable planning task (Haslum, 2006; Haslum & Geffner, 2000; Haslum, Bonet, & Geffner, 2005; Edelkamp, 2001; Helmert, Haslum, & Hoffmann, 2007).

Unfortunately, the palette of known tractable fragments of planning is still very limited, and the situation is even more severe for tractable cost-optimal planning. To the best of our knowledge, no

more than a few non-trivial fragments of cost-optimal planning are known to be tractable. While there is no difference in the theoretical complexity of satisficing and cost-optimal planning in the general case (Bylander, 1994), many classical planning domains are provably easy to solve but hard to solve optimally (Helmert, 2003). Practice also provides clear evidence for the strikingly different scalability of satisficing and cost-optimal general-purpose planners (Hoffmann & Edelkamp, 2005).

In this work we exploit computational tractability to boost general cost-optimal planning. In general, planning algorithms perform reachability analysis in large-scale state models that are implicitly described in a concise manner via some intuitive declarative language (Russell & Norvig, 2004; Ghallab, Nau, & Traverso, 2004). The use of such a language saves the planning algorithms from being restricted to tasks from a certain domain, e.g., a domain of transportation tasks, but rather allows a wide spectrum of tasks describable in that language. Over the years several such languages were developed and studied; the most popular examples are the classical STRIPS language (Fikes & Nilsson, 1971), the PDDL language (McDermott, Ghallab, Howe, Kambhampati, Knoblock, Ram, Veloso, Weld, & Wilkins, 1998), and the SAS$^+$ language (Bäckström & Klein, 1991; Bäckström & Nebel, 1995). Each such language operates with some notion of *initial state*, *actions*, and *goal*. The initial state describes the state of the world at the starting point, actions allow us to move from one world state to another, and the goal describes the desirable outcome. Together they implicitly define a state model, which explicitly captures the world states and the moves between them (as defined by actions), the initial state, and the set of goal states. In the world of sequential planning, the reachability analysis in such a state model corresponds to the search for a path from the initial state to some goal state. This path can be viewed as a *sequence* of corresponding actions, generally named a *plan*. The *cost* of such a plan is defined as the sum of the costs of each individual action along this plan.

Though planning tasks have been studied since the early days of artificial intelligence (Allen, Hendler, & Tate, 1990; Chapman, 1987; Dean & Wellman, 1991; Hendler, Tate, & Drummond, 1990; Wilkins, 1984), recent developments have dramatically advanced the field (Geffner, 2002; Ghallab et al., 2004; Weld, 1999). Two main concerns should be addressed in planning, and in particular, in planning as search. The first is the size of the search space, that is, the number of search states examined before a goal state is found. The size of the search space determines the scalability of the search procedure. The second is the quality of the discovered goal-achieving action sequence, which should ideally be as cost-efficient as possible. Both concerns can be addressed by controlling the order in which the search states are examined. The basic idea is to specify a *heuristic function* h from states to scalars, estimating the cost (of the cheapest path) from states to their nearest goal state. The search algorithms then take these heuristic estimates as search guides. In particular, well-known search algorithms such as A* and IDA* explore the search nodes s in a non-decreasing order of $g(s) + h(s)$, where $g(s)$ is the true cost of reaching s from the initial search state (Hart, Nilsson, & Raphael, 1968; Pearl, 1984; Korf, 1985; Korf & Pearl, 1987; Korf, 1998, 1999). If h is *admissible*, that is, it never overestimates the true cost of reaching the nearest goal state, then such search algorithms are guaranteed to provide an optimal plan to the goal.

A useful admissible heuristic function must be accurate and efficiently computable. Improving the accuracy of a heuristic function without substantially worsening the time complexity of computing it usually translates into faster search for optimal solutions. Since the late 1990s, numerous admissible heuristics for domain-independent planning have been proposed and found practically effective, with research in this direction continuously expanding. Most of the heuristic functions are based on one of the following four ideas:

- The idea of *delete-relaxation* introduced several heuristics, such as h^+ (Hoffmann & Nebel, 2001), $h^{\max}$ and h^{add} (Bonet & Geffner, 2001), h^{FF} (Hoffmann & Nebel, 2001), h^{pmax} (Mirkis & Domshlak, 2007), and h^{sa} (Keyder & Geffner, 2008), with h^+ and $h^{\max}$ being the only admissible representatives of this family, while h^+ is in general intractable (Bylander, 1994).

- The h^m family (Haslum & Geffner, 2000) of *critical path* heuristics, with the $h^1 \equiv h^{\max}$ member being closely related to the *delete relaxation* idea, continued the research in this direction.

- In parallel, Edelkamp (2001) adapted the ideas of Culberson and Schaeffer (1998) for domain-independent planning, and introduced the first member of the *abstraction* heuristics family, the pattern database (PDB) heuristic. Several works continued the research in this direction, expanding and generalizing the idea of *abstraction* heuristics (Edelkamp, 2001; Haslum et al., 2005; Haslum, Botea, Helmert, Bonet, & Koenig, 2007; Helmert et al., 2007).

- Recently, Richter, Helmert, and Westphal (2008) introduced an (inadmissible) *landmarks* based heuristic. Shortly after that, several admissible landmark-based heuristics were introduced, all closely related to delete relaxation of the planning tasks.

A closer look at these advances in domain-independent admissible heuristics reveals the main question: What constraints should we relax to obtain an effective approximation of the planning task? In general, an approximation of the planning task can be obtained in several ways. One way is to systematically contract several states to create a single abstract state. Approximations obtained this way are called *homomorphism abstractions*. Most typically, such a state-gluing is obtained by projecting the original task onto a subset of its parameters, as if ignoring the constraints that fall outside the projection. Homomorphisms have been successfully explored in the scope of domain-independent pattern database (PDB) heuristics (Edelkamp, 2001; Haslum et al., 2005, 2007) and more general explicit abstraction heuristics (Helmert et al., 2007). Another way of obtaining an over-approximation is to enrich the reachability by adding edges and states in the state-space transition graph. Such abstractions are typically referred to as *embedding abstractions*. The latter transformation is typically made by abstracting away a certain (either syntacticly or semantically defined) class of constraints over the task's actions. One such embedding abstraction, based on either full or partial ignoring of negative interactions between the actions, provides the foundations for some of the most influential developments in both satisficing and cost-optimal heuristic search planning (Bonet & Geffner, 2001; Hoffmann & Nebel, 2001; Refanidis & Vlahavas, 2001; Gerevini, Saetti, & Serina, 2003; Vidal, 2004; Haslum et al., 2005; Haslum, 2006; Bonet & Geffner, 2006; Hoffmann & Brafman, 2006; Domshlak & Hoffmann, 2006; Zhou & Hansen, 2006; Richter et al., 2008; Helmert & Domshlak, 2009).

Generally speaking, an abstraction of a planning task is given by a mapping $\alpha : S \to S^\alpha$ from the states of the planning task's transition system to the states of some "abstract transition system" such that, for all states $s, s' \in S$, the cost from $\alpha(s)$ to $\alpha(s')$ is upper-bounded by the cost from s to s'. The abstraction heuristic value $h^\alpha(s)$ is then the cost from $\alpha(s)$ to the closest goal state of the abstract transition system. Perhaps the most well-known abstraction heuristics are pattern database (PDB) heuristics, which are based on projecting the planning task onto a subset of its state variables and then *explicitly* searching for optimal plans in the abstract space. Over the years, PDB heuristics have been shown very effective in several hard search problems, including cost-optimal planning (Culberson & Schaeffer, 1998; Edelkamp, 2001; Felner, Korf, &

Hanan, 2004; Haslum et al., 2007). The conceptual limitation of these heuristics, however, is that the size of the abstract space and its dimensionality must be fixed.[1] The more recent merge-and-shrink abstractions (Helmert et al., 2007) generalize PDB heuristics to overcome the latter limitation. Instead of perfectly reflecting just a few state variables, merge-and-shrink abstractions allow all variables to be imperfectly reflected. As demonstrated by the formal and empirical analysis of Helmert et al. (2007), this flexibility often makes the merge-and-shrink abstractions much more effective than PDBs. However, the merge-and-shrink abstract spaces are still searched *explicitly*, and thus they still have to be of fixed size. While quality heuristic estimates can still be obtained for many tasks, this limitation is a critical obstacle for many others.

The main objective of our work is to push the envelope of abstraction heuristics beyond explicit abstractions. In keeping with this agenda, we introduce a principled way to obtain abstraction heuristics that limit neither the dimensionality nor the size of the abstract spaces. The basic idea behind what we call *implicit abstractions* is simple and intuitive: instead of relying on abstract tasks that are easy to solve because they are small, we can rely on abstract tasks belonging to provably tractable fragments of cost-optimal planning. The key point is that, at least theoretically, moving to implicit abstractions does away with the requirement that the abstractions be small. Our contribution, however, is far from being of theoretical interest only:

1. We specify *acyclic causal-graph decompositions*, a general framework for additive implicit abstractions that is based on decomposing the task at hand along its causal graph. In this scope, we study the complexity of cost-optimal planning for tasks specified in terms of propositional and multi-valued state variables, as well as in terms of actions, each of which changes the value of a single variable. In some sense, we continue the line of complexity analysis suggested in (Brafman & Domshlak, 2003), and extend it from satisficing to cost-optimal planning. Our results for the first time provide a dividing line between tractable and intractable such problems (Katz & Domshlak, 2008a).

2. We then introduce a concrete family of additive implicit abstractions, called *fork decompositions*, that are based on two novel fragments of tractable cost-optimal planning (Katz & Domshlak, 2008c). Following the type of analysis suggested by Helmert and Mattmüller (2008), we formally analyze the asymptotic performance ratio of the fork-decomposition heuristics and prove that their worst-case accuracy on selected domains is comparable with that of (even parametric) state-of-the-art admissible heuristics. We then empirically evaluate the accuracy of the fork-decomposition heuristics on a large set of domains from recent planning competitions and show that their accuracy is competitive with the state of the art.

3. The key attraction of explicit abstractions is that state-to-goal costs in the abstract space can be precomputed and stored in memory in a preprocessing phase so that heuristic evaluation during search can be done by a simple lookup. At first view, a necessary condition for pushing computation of the heuristic offline appears to be the small size of the abstract space. We show, however, that an equivalent of the PDB and merge-and-shrink's notion of "database" exists for the fork-decomposition abstractions as well, despite the exponential-size abstract spaces of the latter. These *databased implicit abstractions* are based on a proper partitioning of the heuristic computation into parts that can be shared between search states and parts that must be computed online per state. Our empirical evaluation shows that A^* equipped

[1] This does not necessarily apply to *symbolic* PDBs which, on some tasks, may exponentially reduce the PDB's representation (Edelkamp, 2002; Ball & Holte, 2008).

with the "databased" fork-decomposition heuristics favorably competes with the state of the art of cost-optimal planning (Katz & Domshlak, 2009b).

Of course, as planning is known to be NP-hard even for conservative planning formalisms (Bylander, 1994), no heuristic should be expected to work well in all planning tasks. Moreover, even for a fixed planning task, no tractable heuristic will home in on all the "combinatorics" of the task at hand. The promise, however, is that different heuristics could target different sources of the planning complexity, and composing a set of heuristics to exploit their individual strengths could allow a larger range of planning tasks to be solved as well as each individual task to be solved more efficiently.

Our additional contribution is to the fundamental question of how one should better compose a set of admissible heuristics. One of the well-known and heavily-used properties of admissible heuristics is that taking the maximum of their values maximizes informativeness while preserving admissibility. A more recent, alternative approach to composing a set of admissible heuristics corresponds to carefully separating the information used by the different heuristics in the set so that their values could be summed instead of maximized over. This direction was first exploited in devising domain-specific heuristics, and more recently in works on additive pattern database (PDB) heuristics (Edelkamp, 2001; Felner et al., 2004; Haslum et al., 2007) and constrained PDBs and m-reachability heuristics (Haslum et al., 2005).

The basic idea underlying all these *additive heuristic ensembles* is elegantly simple: for each planning task's action a, if it can possibly be counted by more than one heuristic in the ensemble, then one should ensure that the cumulative counting of the cost of a does not exceed its true cost in the original task. Such *action cost partitioning* was originally achieved by accounting for the whole cost of each action in computing a single heuristic in the ensemble, while ignoring the cost of that action in computing all the other heuristics in the ensemble (Edelkamp, 2001; Felner et al., 2004; Haslum et al., 2005). Recently, this "all-in-one/nothing-in-rest" action-cost partitioning has been generalized to *arbitrary* partitioning of the action cost among the heuristics in the ensemble (Katz & Domshlak, 2007a, 2008c; Yang, Culberson, & Holte, 2007; Yang, Culberson, Holte, Zahavi, & Felner, 2008).

The great flexibility of additive heuristic ensembles, however, is a mixed blessing. For better or for worse, the methodology of taking the maximum over the values provided by an arbitrary set of independently constructed admissible heuristics is entirely nonparametric. By contrast, switching to additive heuristic ensembles requires *selecting an action-cost partitioning scheme*, and this decision problem poses a number of computational challenges:

- The space of alternative action-cost partitions is infinite as the cost of each action can be partitioned into an arbitrary set of nonnegative real numbers, the sum of which does not exceed the cost of that action.

- At least in domain-independent planning, what we need is a fully unsupervised decision process.

- Last but not least, the relative quality of each action-cost partition (in terms of the accuracy of the resulting additive heuristic) may vary dramatically between the examined search states. Hence, the choice of the action-cost partitioning scheme should ultimately be a function of the search state in question.

These concerns may explain why all previous works on both domain-specific and domain-independent additive heuristic ensembles adopt this or another ad hoc, *fixed* choice of action-cost partition. Consequently, all the reported empirical comparative evaluations of various max-based and additive heuristic ensembles are inconclusive—for some search states along the search process, the (pre-selected) additive heuristic was found to dominate the max-combination, while for the other states the opposite was the case. In the context of domain-specific additive PDBs, Yang et al. (2007) conclude that "determining which abstractions [here: action-cost partitioning schemes] will produce additives that are better than max over standards is still a big research issue."

Focusing on abstraction heuristics, our contribution here is precisely in addressing the problem of choosing the right action-cost partitioning over a given set of heuristics:

1. We provide a procedure that, given (i) a classical planning task Π, (ii) a forward-search state s of Π, and (iii) a set of heuristics based on abstractions of Π, derives an *optimal action-cost partition for* s, that is, a partition that maximizes the heuristic estimate of that state. The procedure is *fully unsupervised*, and is based on a linear programming formulation of that optimization problem.

2. We show that the time complexity of our procedure is *polynomial* for arbitrary sets of *all* abstraction-based heuristic functions with which we are familiar. Such "procedure-friendly" heuristics include PDBs (Edelkamp, 2001; Yang et al., 2007), constrained PDBs (Haslum et al., 2005), merge-and-shrink abstractions (Helmert et al., 2007), fork-decomposition implicit abstractions (Katz & Domshlak, 2008c), and implicit abstractions based on tractable constraint optimization over tree-shaped constraint networks (Katz & Domshlak, 2008a). Note that the estimate provided by a max-based ensemble corresponds to the estimate provided by the respective additive ensemble under *some* action-cost partitioning. Thus, by finding an *optimal* action-cost partition, we provide a formally complete answer to the aforementioned question of "to add or not to add" in the context of abstraction heuristics.

Taking the fork-decomposition abstractions as a case study, we evaluate the empirical effectiveness of switching from ad hoc to optimal additive composition. Our evaluation on a wide range of International Planning Competition (IPC) benchmarks shows a substantial reduction in the number of nodes expanded by the A^* algorithm. However, in the standard time-bounded setting, this reduction in expanded nodes is typically negatively balanced by the much more expensive per-node computation of the optimal additive heuristic. To overcome this pitfall without forfeiting the promise of optimized action cost partitions altogether, we suggest that optimal action cost partitions be derived only with respect to a subset of evaluated nodes. We examine this approach empirically in its extreme setting where only one optimal action cost partition is computed per planning task: the one that is optimal for the task's initial state. This action cost partition is then used for all the states evaluated by the A^* algorithm. Our experiments show that even such a conservative use of optimization results in substantial improvement over the same heuristic-search planner relying on an ad hoc action cost partition (Katz & Domshlak, 2010).

The rest of the dissertation is organized as follows.

- Chapter 2 presents the essential definitions and constructs used in this work and gives a brief introduction to heuristics and heuristic-search planning, with a focus on the abstraction heuristics.

- Chapter 3 presents a study of the computational tractability of cost-optimal planning. Numerous new tractable classes of cost-optimal planning are presented (Katz & Domshlak, 2007b, 2007a, 2008a). The results are based on exploiting several structural and syntactic characteristics of planning tasks such as the structure of their causal graphs.

- Chapter 4 introduces the basic idea and theory behind *implicit abstractions* (Katz & Domshlak, 2007b, 2007a, 2008c, 2009a). The complexity study presented in Chapter 3 plays a major role in implementing this theory. We introduce a concrete family of additive implicit abstractions, called *fork decompositions*, which are based on two novel fragments of tractable cost-optimal planning introduced in Chapter 3. Likewise, we develop a "databased" approach to these implicit abstractions and show that they favorably compete with the state of the art of cost-optimal planning.

- Chapter 5 presents the action-cost partitioning framework for composing abstraction-based heuristics. Such a composition allows us to use multiple abstractions and even multiple types of abstractions together to obtain more informative estimates. In particular, we show that this framework allows for obtaining an *optimal* cost partitioning in polynomial time, and we show that all abstraction-based heuristics known to us fit into this framework. In addition, we discuss the possibility of relinquishing the optimality of the composed heuristic for the sake of speed, and show that it favorably competes with the state of the art of cost-optimal planning.

- Chapter 6 summarizes our work and outlines some current and future work prospects.

For better flow and readability, several detailed constructions were moved to appendices.

- Appendix A describes in detail the complexity results presented in Chapter 3, and

- Appendix B describes the empirical evaluation in detail for several heuristics and cost partition schemes suggested in Chapters 4-5.

Chapter 2

Classical Planning, Heuristic Search, and Abstractions

2.1 Planning Tasks and Abstractions

The development of general problem solvers (GPS) has been one of the main goals in operations research, as well as in the artificial intelligence branch of computer science. A general problem solver is a program that accepts high-level descriptions of problems and automatically computes their solution. The practical motivation for such solvers is that modeling problems at a high level is most often simpler than procedurally encoding their solutions. Thus, an effective GPS tool can be very useful in practice.

A general problem solver must provide the user with a suitable *general modeling language* for describing problems, and *general algorithms* for solving them. While obtaining a solution using a GPS might be somewhat slower than obtaining it by more specialized methods, this should not necessarily be the case. Most importantly, the GPS approach typically pays off if developing, implementing, and testing the specialized solution is just too cumbersome, time-consuming, and cost-inefficient (Muscettola, Nayak, Pell, & Williams, 1998). The scope of a general problem solver can be defined by three elements:

1. *mathematical models* for making the different planning problems precise,
2. *representation languages* for describing problems conveniently, and
3. *algorithms* for solving these problems effectively, often making use of information available in their representation.

In this work we focus on general solvers for planning problems (Ghallab et al., 2004).

Probably the most fundamental class of planning tasks is that of *classical planning*, corresponding to state models with deterministic actions and complete information. Formally, such a **state model** is a tuple $\mathcal{S} = \langle S, s^I, S^G, A, Tr, cost \rangle$ where

- S is a finite set of states, $s^I \in S$ is the initial state, and $S^G \subseteq S$ is a set of alternative goal states,
- A is a finite set of actions, and for each $s \in S$, $A(s) \subseteq A$ denotes the set of all actions applicable in s,

Objects			Action	SAS$^+$ representation
$v \in \{c_1, c_2\}$,	$\{l, l'\} \subseteq \{A, B, C, D\}$		$Move(v, l, l')$	$\langle\{v = l\}, \{v = l'\}\rangle$
$v = c_3$,	$\{l, l'\} \subseteq \{E, F, G\}$			
$v = t$,	$\{l, l'\} \subseteq \{D, E\}$			
$v \in \{c_1, c_2, c_3\}$,	$l \in \{A, B, C, D, E, F, G\}$,	$p \in \{p_1, p_2\}$	$Load(p, v, l)$	$\langle\{p = l, v = l\}, \{p = v\}\rangle$
$v = t$,	$l \in \{D, E\}$,	$p \in \{p_1, p_2\}$	$Unload(p, v, l)$	$\langle\{p = v, v = l\}, \{p = l\}\rangle$

Table 2.1: The actions of the Logistics-style example planning task and their SAS$^+$ representation. The actions $Move(v, l, l')$ stand for moving the vehicle v from the location l to the location l'. The actions $Load(p, v, l)$ stand for loading the package p into the vehicle v from the location l. The actions $Unload(p, v, l)$ stand for unloading the package p from the vehicle v to the location l.

- $Tr : S \times A \to S$ is a transition function, such that if $a \in A(s)$, then $Tr(s, a)$ specifies the state obtained from applying a in s, otherwise (a convention is that) $Tr(s, a) = s$,

- $cost : S \times A^* \to \mathbb{R}^{0+}$ captures the cost of applying an action sequence in a state.

A solution (= plan) for such a state model $\mathcal{S}$ is a sequence of actions $\rho = \langle a_1, \ldots, a_m\rangle$ that generates a sequence of states $s_0, \ldots, s_m$ such that, for $0 \leq i < m$, a_{i+1} is applicable in s_i, $Tr(s_i, a_{i+1}) = s_{i+1}$, and $s_m \in S^G$. A plan ρ is cost-optimal if $cost(s^I, \rho)$ is minimal over all plans for $\mathcal{S}$.

This model provides a mathematical characterization of the basic planning problems, and their solutions. It does not, however, provide a convenient *language* for encoding planning problems. This is because the explicit characterization of the state space and state transitions is feasible only in small problems. In large problems, the state space and state transitions need to be represented *implicitly* in a logical *action language*, normally through a set of (state) variables and action rules. A good action language is one that supports compact encodings of the models of interest (Fikes & Nilsson, 1971). For instance, a popular language for a classical-planning model as above is the **sas$^+$ language** (Bäckström & Klein, 1991; Bäckström & Nebel, 1995) in which a planning task is given by a quintuple $\Pi = \langle V, A, I, G, cost\rangle$, where:

- V is a set of *state variables*, with each $v \in V$ being associated with a finite domain $\mathcal{D}(v)$. For a subset of variables $V' \subseteq V$, we denote the set of assignments to V' by $\mathcal{D}(V') = \times_{v \in V'} \mathcal{D}(v)$. A complete assignment to V is called a *state*, and $S = \mathcal{D}(V)$ is the *state space* of Π. I is an *initial state*. The *goal* G is a partial assignment to V; a state s is a *goal state* iff $G \subseteq s$.

- A is a finite set of *actions*. Each action a is a pair $\langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ of partial assignments to V called *preconditions* and *effects*, respectively. By $A_v \subseteq A$ we denote the actions affecting the value of v. $cost : A \to \mathbb{R}^{0+}$ is a real-valued, nonnegative *action cost* function.

To illustrate various constructs, we use a slight variation of a Logistics-style example from Helmert

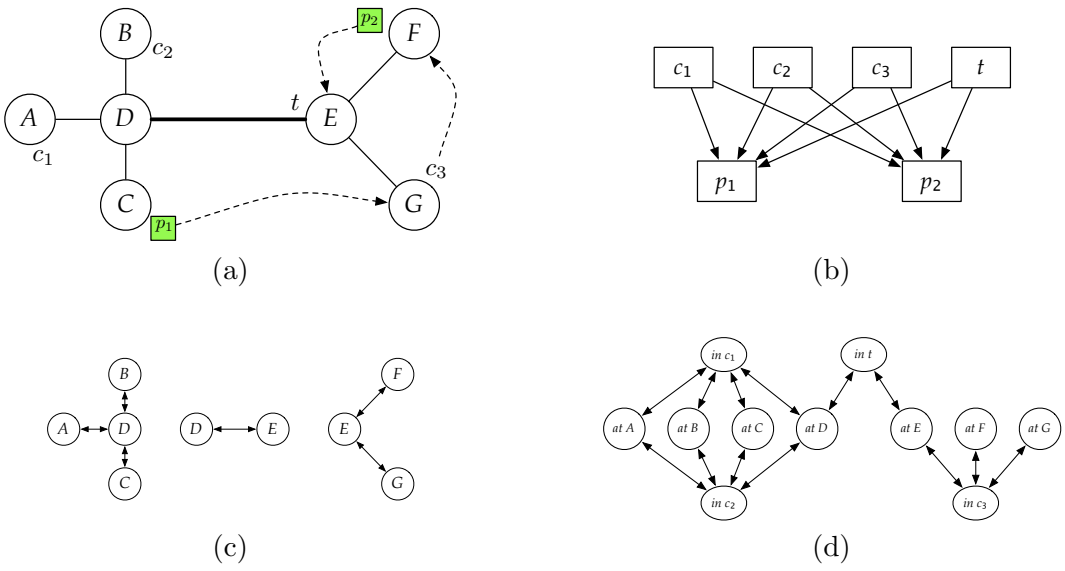


Figure 2.1: Logistics-style example adapted from Helmert (2006) and illustrated in (a). The goal is to deliver p_1 from C to G and p_2 from F to E using the cars c_1, c_2, c_3 and truck t, making sure that c_3 ends up at F. The cars may only use city roads (thin edges); the truck may only use the highway (thick edge). Figures (b), (c), and (d) depict, respectively, the causal graph of the problem, the domain transition graphs (labels omitted) of c_1 and c_2 (left), t (center), and c_3 (right), and the identical domain transition graphs of of p_1 and p_2.

(2006). This example is depicted in Figure 2.1a, and in SAS$^+$ it has

$$\begin{aligned} V &= \{p_1, p_2, c_1, c_2, c_3, t\} \\ \mathcal{D}(p_1) &= \mathcal{D}(p_2) = \{A, B, C, D, E, F, G, c_1, c_2, c_3, t\} \\ \mathcal{D}(c_1) &= \mathcal{D}(c_2) = \{A, B, C, D\} \\ \mathcal{D}(c_3) &= \{E, F, G\} \\ \mathcal{D}(t) &= \{D, E\} \\ I &= \{p_1{=}C, p_2{=}F, t{=}E, c_1{=}A, c_2{=}B, c_3{=}G\} \\ G &= \{p_1{=}G, p_2{=}E, c_3{=}F\}, \end{aligned}$$

and actions (see Table 2.1) corresponding to all possible loads and unloads, as well as single-segment movements of the vehicles. For instance, if action a captures loading p_1 into c_1 at C, then $\mathsf{pre}(a) = \{p_1{=}C, c_1{=}C\}$, and $\mathsf{eff}(a) = \{p_1{=}c_1\}$. In what follows, we assume that all actions in the example have unit cost.

We now provide some helpful notations. For a partial assignment p, $\mathcal{V}(p) \subseteq V$ denotes the subset of state variables instantiated by p. In turn, for any $V' \subseteq \mathcal{V}(p)$, by $p[V']$ we denote the value of V' in p; if $V' = \{v\}$ is a singleton, we use $p[v]$ for $p[V']$. For any sequence of actions ρ and variable $v \in V$, by $\rho{\downarrow_v}$ we denote the restriction of ρ to actions changing the value of v, that is, $\rho{\downarrow_v}$ is the maximal subsequence of ρ consisting only of actions in A_v.

The semantics of actions and action sequences in SAS$^+$ planning tasks are as follows. An action a is applicable in a state s iff $s[v] = \mathsf{pre}(a)[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$. Applying a changes the value of

$v \in \mathcal{V}(\mathsf{eff}(a))$ to $\mathsf{eff}(a)[v]$. The resulting state is denoted by $s[\![a]\!]$; by $s[\![\langle a_1, \ldots, a_k \rangle]\!]$ we denote the state obtained from sequential application of the (respectively applicable) actions $a_1, \ldots, a_k$ starting at state s. Such an action sequence is an *s-plan* if $G \subseteq s[\![\langle a_1, \ldots, a_k \rangle]\!]$, and it is a *cost-optimal* (or, in what follows, *optimal*) s-plan if the sum of its action costs is minimal among all s-plans. The purpose of (optimal) planning is finding an (optimal) I-plan. For a pair of states $s_1, s_2 \in S$, by $cost(s_1, s_2)$ we refer to the cost of a cheapest action sequence taking us from s_1 to s_2 in the transition system induced by Π; $h^*(s) = \min_{s' \supseteq G} cost(s, s')$ is the customary notation for the cost of the optimal s-plan in Π. For each action $a \in A$ and each variable $v \in V$, by $\mathsf{Pre}(a)[v] \subseteq \mathcal{D}(v)$ we denote the set of values of v which do not preclude applying a, that is

$$\mathsf{Pre}(a)[v] = \begin{cases} \{\mathsf{pre}(a)[v]\}, & v \in \mathcal{V}(\mathsf{pre}(a)) \\ \mathcal{D}(v), & \text{otherwise} \end{cases}.$$

In this work we focus on *cost-optimal* (also known as *sequentially optimal*) planning, in which the goal is to find a plan $\rho \in A^*$ for task Π minimizing $\sum_{a \in \rho} cost(a)$.

We now proceed with formalizing a few essential constructs. The semantics of a planning task Π as a whole is given by its induced state-transition model, often called its *transition graph*. Searching in this transition graph corresponds to forward state-space search. For our constructions later on, we distinguish between the actual edge weighted transition graph, and its weights-omitted, qualitative skeleton, which we call the *transition-graph structure*. Informally, transition-graph structures capture the dynamics of the planning tasks, while transition graphs associate the dynamics with "performance measures."

Definition 1 *A* **transition-graph structure** *(or* **TG-structure***, for short) is a quintuple* $\mathcal{T} = (S, L, Tr, s^I, S^G)$ *where* S *is a finite set of* states*,* L *is a finite set of* transition labels*,* $Tr : S \times L \to S$ *is a (labeled)* transition function*,* $s^I \in S$ *is an* initial state*, and* $S^G \subseteq S$ *is a set of* goal states*. Any path from* s^I *to* S^G *is a* **plan** *for* $\mathcal{T}$*.*

Definition 2 *A* **transition graph** *is a pair* $\langle \mathcal{T}, \varpi \rangle$ *where* $\mathcal{T}$ *is a TG-structure with labels* L*, and* $\varpi : L \to \mathbb{R}^{0+}$ *is a* transition cost function*. For a state* $s \in S$ *and a subset of states* $S' \subseteq S$ *in* $\mathcal{T}$*,* $cost(s, S')$ *is the* **cost** *(of a cheapest with respect to* ϖ *path) from* s *to a state in* S' *along the transitions of* $\mathcal{T}$*; if no state in* S' *is reachable from* s*, then we have* $cost(s, S') = \infty$*. Plans for* $\langle \mathcal{T}, \varpi \rangle$ *are simply the plans for its TG-structure* $\mathcal{T}$*, and cheapest such plans are called* **optimal plans***.*

The states of the TG-structure $\mathcal{T}(\Pi)$ induced by a planning task $\Pi = \langle V, A, I, G, cost \rangle$ are the states of Π. The transition labels of $\mathcal{T}(\Pi)$ are the actions A, and $Tr(s, a) = s[\![a]\!]$ if a is applicable in s, and $Tr(s, a) = s$ otherwise. The actual transition graph induced by Π is $\langle \mathcal{T}(\Pi), cost \rangle$.

We now proceed with formally specifying the notion of abstraction. Our definition of abstraction resembles this of Prieditis (1993), and right from the beginning we specify a more general notion of *additive abstraction*. Informally, by additive abstraction we refer to a set of abstractions interconstrained by a requirement to *jointly* not overestimate the transition-path costs in the abstracted transition graph.

Definition 3 *An* **additive abstraction** *of a transition graph* $\langle \mathcal{T}, \varpi \rangle$ *with structure* $\mathcal{T} = (S, L, Tr, s^I, S^G)$ *is a set of pairs* $\mathcal{A} = \{\langle \langle \mathcal{T}_i, \varpi_i \rangle, \alpha_i \rangle\}_{i=1}^k$ *where, for* $1 \leq i \leq k$*,*

- $\langle \mathcal{T}_i, \varpi_i \rangle$ *is a transition graph with structure* $\mathcal{T}_i = (S_i, L_i, Tr_i, s_i^I, S_i^G)$,
- $\alpha_i : S \to S_i$ *is a function, called* abstraction mapping*, such that*
  - $\alpha_i(s^I) = s_i^I$, $\alpha_i(s) \in S_i^G$ *for all* $s \in S^G$*, and,*
  - *for all pairs of states* $s, s' \in S$*, it holds that*

$$\sum_{i=1}^{k} cost(\alpha_i(s), \alpha_i(s')) \leq cost(s, s'). \tag{2.1}$$

A few words on why we use *this* particular notion of abstraction. The term "abstraction" is usually associated with simplifying the original system, reducing and factoring out details less crucial in the given context. Which details can be reduced and which should be preserved depends, of course, on the context. For instance, in the context of formal verification, the abstract transition graphs are required not to decrease the reachability between the states; that is, if there is a path from s to s' in the original transition graph, then there should be a path from $\alpha(s)$ to $\alpha(s')$ in the abstract transition graph (Clarke, Grumberg, & Peled, 1999). In addition, the reachability should also be increased as little as possible. Beyond that, the precise relationship between the path costs in the original and abstract transition graphs is only of secondary importance, if it is important at all. By contrast, when abstractions are designed to induce admissible heuristic functions for heuristic search, the relationship between the path costs as captured by Eq. 2.1 is what must be obeyed. However, requirements above and beyond the general requirement of Eq. 2.1 not to overestimate the path costs between the states are unnecessary. Hence, in particular, Definition 3 generalizes the notion of abstraction by Helmert et al. (2007) by replacing the condition of preserving individual transitions and their labels, that is, $(\alpha(s), l) \mapsto \alpha(s')$ if $(s, l) \mapsto s'$, with a weaker condition stated in Eq. 2.1. The reader, of course, may well ask whether the generality of the condition in Eq. 2.1 beyond the condition of Helmert et al. (2007) really delivers any practical gain. Later we show that the answer to this question is affirmative.

Finally, important roles in what follows are played by a pair of standard graphical structures induced by planning tasks.

Definition 4 *The* **causal graph** $CG(\Pi)$ *of* Π *is a directed graph over nodes* V*. An arc* (v, v') *is in* $CG(\Pi)$ *iff* $v \neq v'$ *and there exists an action* $a \in A$ *such that* $(v, v') \in \mathcal{V}(\mathsf{eff}(a)) \cup \mathcal{V}(\mathsf{pre}(a)) \times \mathcal{V}(\mathsf{eff}(a))$*. In this case, we say that* (v, v') *is* induced *by* a*. By* $\mathsf{succ}(v)$ *and* $\mathsf{pred}(v)$ *we respectively denote the sets of immediate successors and predecessors of* v *in* $CG(\Pi)$*.*

Informally, the causal graph of a planning task captures the dependency relation between the variables, as defined by the task's actions. The causal graph of our running example is depicted in Figure 2.1b.

Definition 5 *The* **domain transition graph** $DTG(v, \Pi)$ *of a variable* $v \in V$ *is an arc-labeled directed graph over the nodes* $\mathcal{D}(v)$ *such that an arc* (ϑ, ϑ')*, labeled with* $\mathsf{pre}(a)[V \setminus \{v\}]$ *and* $cost(a)$*, belongs to* $DTG(v, \Pi)$ *iff both* $\mathsf{eff}(a)[v] = \vartheta'$ *and* $\vartheta \in \mathsf{Pre}(a)[v]$*.*

The domain transition graph of variable v captures the possible value changes of that variable. Figures 2.1c-d depict the (label-omitted) domain transition graphs of the variables in our running example.

2.2 Heuristic Functions and Abstractions

As mentioned above, planning as heuristic search is one of the major tools for solving planning tasks. Heuristic search consists of two components, namely the informed search algorithm and the heuristic function. Although these components should be properly combined, they may be addressed separately. As this work focuses on heuristic functions, we only briefly mention the informed search algorithms.

A general informed search algorithm is best-first search, which expands nodes based on an evaluation function f that measures the cost of reaching the goal. Thus, the node with the lowest f-value is selected for expansion. Best-first search can be implemented using a priority queue, which maintains the search frontier in ascending order of f-values. The most widely-known member of the family of best-first search algorithms is the A^* search algorithm (Hart et al., 1968), which uses the evaluation function $f = g + h$ with g being the cost of reaching the node and h being the heuristic function. If the heuristic function h is *admissible*, that is, never overestimates the true cost of reaching the goal, A^* is guaranteed to return an optimal solution (Dechter & Pearl, 1985).

2.2.1 Admissible Heuristics and Heuristic Ensembles

Heuristic functions are used by informed-search procedures to estimate the cost (of the cheapest path) from a search node to the nearest goal node. Our focus here is on state-dependent, admissible abstraction heuristics. A heuristic is *state-dependent* if its estimate for a search node depends only on the planning task state associated with that node, that is, $h : S \to \mathbb{R}^{0+} \cup \{\infty\}$. Most heuristics in use these days are state-dependent (though see, e.g., Richter et al. (2008) and Karpas and Domshlak (2009) for a different case). A heuristic h is *admissible* if $h(s) \leq h^*(s)$ for all states s. If h_1 and h_2 are two admissible heuristics, and $h_2(s) \leq h_1(s)$ for all states s, we say that h_1 *dominates* h_2.

A useful heuristic function must be accurate as well as efficiently computable. Improving the accuracy of a heuristic function without substantially worsening the time complexity of computing it usually translates into faster search for optimal solutions. During the last decade, numerous computational ideas have evolved into new admissible heuristics for classical planning; these include the delete-relaxing max heuristic $h_{\max}$ (Bonet & Geffner, 2001), critical path heuristics h^m (Haslum & Geffner, 2000), landmark heuristics h^{L}, h^{LA} (Karpas & Domshlak, 2009) and $h^{\text{LM-cut}}$ (Helmert & Domshlak, 2009), and abstraction heuristics such as pattern databases (Edelkamp, 2001) and merge-and-shrink (Helmert et al., 2007).

For any set of admissible heuristics $h_1, \ldots, h_m$, their pointwise maximum is always an admissible heuristic, dominating each individual heuristic in the set. This property of admissible heuristics is widely used in the context of optimal search. For some sets of admissible heuristics, however, their pointwise sum is also admissible and dominates their pointwise maximum. Many recent works on cost-optimal planning are based on *additive ensembles of admissible heuristics*, and these include critical-path (Haslum et al., 2005; Coles, Fox, Long, & Smith, 2008), pattern database (Edelkamp, 2001; Haslum et al., 2007), and landmark (Karpas & Domshlak, 2009; Helmert & Domshlak, 2009) heuristics.

Cost partitioning offers a flexible way of additively combining different heuristic estimates while guaranteeing admissibility of the resulting combination. It subsumes earlier admissibility criteria for additive pattern database heuristics by Edelkamp (2001) and for general admissible heuristics by Haslum et al. (2007). Of course, different cost partitions lead to additive heuristics of different

quality, and thus the question of how to automatically derive a good cost partition is of interest. This is precisely the question considered in what follows.

2.2.2 Abstraction Heuristics

Our main focus in this work is on abstraction heuristics. In the most general form an *abstraction heuristic* is defined as follows.

Definition 6 *Let Π be a planning task over states S, and let $\mathcal{A} = \{\langle\langle \mathcal{T}_i, \varpi_i\rangle, \alpha_i\rangle\}_{i=1}^{k}$ be an additive abstraction of the transition graph $\mathcal{T}(\Pi)$. If $k = O(poly(||\Pi||))$ and, for all states $s \in S$ and all $1 \leq i \leq k$, the cost $cost(\alpha_i(s), S_i^G)$ in $\langle \mathcal{T}_i, \varpi_i\rangle$ is computable in time $O(poly(||\Pi||))$, then $h^{\mathcal{A}}(s) = \sum_{i=1}^{k} cost(\alpha_i(s), S_i^G)$ is an* **abstraction heuristic** *function for Π.*

Note that admissibility of $h^{\mathcal{A}}$ is implied by the cost conservation condition of Eq. 2.1. To further illustrate the connection between abstractions and admissible heuristics, consider three well-known mechanisms for devising admissible planning heuristics: delete relaxation (Bonet & Geffner, 2001), critical-path relaxation (Haslum & Geffner, 2000),[1] and pattern database heuristics (Edelkamp, 2001).

First, while typically not considered this way, the delete relaxation of a planning task $\Pi = \langle V, A, I, G, cost\rangle$ does correspond to an abstraction $\langle \mathcal{T}_+ = (S_+, L_+, Tr_+, s_+^I, S_+^G, \varpi_+), \alpha_+\rangle$ of the transition graph $\mathcal{T}(\Pi)$. Assuming unique naming of the variable values in Π and denoting $\mathcal{D}_+ = \bigcup_{v \in V} \mathcal{D}(v)$, we have the abstract states S_+ being the power-set of $\mathcal{D}_+$, and the labels $L_+ = \{a, a_+ \mid a \in A\}$. The transitions come from two sources: for each abstract state $s_+ \in S_+$ and each original action $a \in A$ applicable in s_+, we have both $Tr_+(s_+, a) = s_+[\![a]\!]$ and $Tr_+(s_+, a_+) = s_+ \cup \mathsf{eff}(a)$. With a minor abuse of notation, the initial state and the goal states of the abstraction are $s_+^I = I$ and $S_+^G = \{s_+ \in S_+ \mid s_+ \supseteq G\}$, and the abstraction mapping α_+ is simply the identity function. It is easy to show that, for any state s of our planning task Π, we have $cost(\alpha_+(s), S_+^G) = h^+(s)$, where $h^+(s)$ is the delete-relaxation estimate of the cost from s to the goal. As an aside, we note that this "delete-relaxation abstraction" $\langle \mathcal{T}_+, \alpha_+\rangle$ in particular exemplifies that nothing in Definition 3 requires the size of the abstract state space to be limited by the size of the original state space. In any event, however, the abstraction $\langle \mathcal{T}_+, \alpha_+\rangle$ does *not* induce a heuristic in terms of Definition 6 because computing $h^+(s)$ is known to be NP-hard (Bylander, 1994).

The situation for critical-path relaxation is exactly the opposite. While computing the corresponding family of admissible estimates h^m is polynomial-time for any fixed m, this computation is not based on computing the shortest paths in an abstraction of the planning task. The state graph over which h^m is computed is an AND/OR-graph (and not an OR-graph such as transition graphs), and the actual computation of h^m corresponds to computing a critical tree (and not a shortest path) to the goal. To the best of our knowledge, the precise relation between critical path and abstraction heuristics is currently an open question (Helmert & Domshlak, 2009).

2.2.3 Explicit Abstractions

Overall, the only abstraction heuristics in the planning toolbox these days appear to be the *explicit homomorphism abstractions*, whose best-known representative is probably the *pattern database (PDB)* heuristics. Given a planning task Π over state variables V, a PDB heuristic is based on

[1] We assume the reader is familiar with these two relaxations. If not, their discussion here can be safely skipped.

projecting Π onto a subset of its variables $V^\alpha \subseteq V$. Such a homomorphism abstraction α maps two states $s_1, s_2 \in S$ into the same abstract state iff $s_1[V^\alpha] = s_2[V^\alpha]$. Inspired by the (similarly named) domain-specific heuristics for search problems such as (n^2-1)-puzzles or Rubik's Cube (Culberson & Schaeffer, 1998; Hernadvölgyi & Holte, 1999; Felner et al., 2004), PDB heuristics have been successfully exploited in domain-independent planning as well (Edelkamp, 2001, 2002; Haslum et al., 2007). The key decision in constructing PDBs is what sets of variables the problem is projected to (Edelkamp, 2006; Haslum et al., 2007). However, apart from that need to automatically select good projections, the two limitations of PDB heuristics are the size of the abstract space S^α and its dimensionality. First, the number of abstract states should be small enough to allow reachability analysis in S^α by exhaustive search. Moreover, an $O(1)$ bound on $|S^\alpha|$ is typically set explicitly to fit the time and memory limitations of the system. Second, since PDB abstractions are projections, the explicit constraint on $|S^\alpha|$ implies a fixed-dimensionality constraint $|V^\alpha| = O(1)$. In planning tasks with, informally, many alternative resources, this limitation is a pitfall. For instance, suppose $\{\Pi_i\}_{i=1}^\infty$ is a sequence of Logistics problems of growing size with $|V_i| = i$. If each package in Π_i can be transported by some $\Theta(i)$ vehicles, then starting from some i, h^α will not account *at all* for movements of vehicles essential for solving Π_i (Helmert & Mattmüller, 2008).

With the goal of preserving the attractiveness of the PDB heuristic while eliminating the bottleneck of fixed dimensionality, Helmert et al. (2007) have generalized the methodology of Dräger, Finkbeiner, and Podelski (2006) and introduced the so called *merge-and-shrink (MS)* abstractions for planning. MS abstractions are homomorphisms that generalize PDB abstractions by allowing for more flexibility in selecting the pairs of states to be contracted. The problem's state space is viewed as the synchronized product of its projections onto the single state variables. Starting with all such "atomic" abstractions, this product can be computed by iteratively composing two abstract spaces, replacing them with their product. While in a PDB the size of the abstract space S^α is controlled by limiting the number of product compositions, in MS abstractions it is controlled by interleaving the iterative composition of projections with abstraction of the partial composites. Helmert et al. (2007) have proposed a concrete strategy for this interleaved abstraction/refinement scheme and empirically demonstrated the power of the merge-and-shrink abstraction heuristics. Like PDBs, however, MS abstractions are explicit abstractions, and thus computing their heuristic values is also based on explicitly searching for optimal plans in the abstract spaces. Hence, while merge-and-shrink abstractions escape the fixed-dimensionality constraint of PDBs, the constraint on the abstract space to be of a fixed size still holds.

Chapter 3

Tractable Fragments of Cost-Optimal Planning

Unfortunately, the palette of known tractable fragments of planning is still very limited, and the situation is even more severe for tractable optimal planning. To the best of our knowledge, just a few non-trivial fragments of optimal planning are known to be tractable. Despite the identical theoretical complexity of regular and optimal planning in the general case (Bylander, 1994), specific planning domains mostly have different complexity (Helmert, 2003). Clear evidence for strikingly different scalability of satisficing and optimal general-purpose planners can be seen in the results of the International Planning Competitions (Hoffmann & Edelkamp, 2005).

In this work we show that the search for new islands of tractability in optimal classical planning is far from exhausted. Specifically, we study the complexity of optimal planning for problems specified in terms of propositional state variables and actions, each of which changes the value of a single variable. In some sense, we continue the line of complexity analysis suggested in Brafman and Domshlak (2003), and extend it from satisficing to optimal planning. Our results for the first time provide a dividing line between tractable and intractable such problems.

Definition 7 *A* SAS$^+$ *task* $\Pi = \langle V, A, I, G, cost \rangle$ *belongs to the* **UB** *(***U***nary-effect,* **B***inary-valued) fragment iff all the state variables in* V *are binary-valued, and each action changes the value of exactly one variable, that is, for all* $a \in A$*, we have* $|\mathsf{eff}(a)| = 1$*.*

Different sub-fragments of UB can be defined by placing syntactic and structural restrictions on the action sets of the problems. For instance, Bylander (1994) shows that planning in UB domains where each action is restricted to have only positive preconditions is tractable, yet optimal planning for this UB fragment is hard. In general, the seminal works of Bylander (1994) and Erol, Nau, and Subrahmanian (1995) indicate that extremely severe syntactic restrictions on single actions are required to guarantee tractability, or even membership in NP. Bäckström and Klein (1991) consider syntactic restrictions of a more global nature, and show that UB planning is tractable if the preconditions of any two actions do not require different values for non-changing variables, and no two actions have the same effect. Interestingly, this fragment of UB, known as **PUBS**, remains tractable for optimal planning as well. While the characterizing properties of **PUBS** are very restrictive, this result of Bäckström and Klein is an important milestone in planning tractability research.

Given the limitations of syntactic restrictions observed by Bylander (1994), Erol et al. (1995),

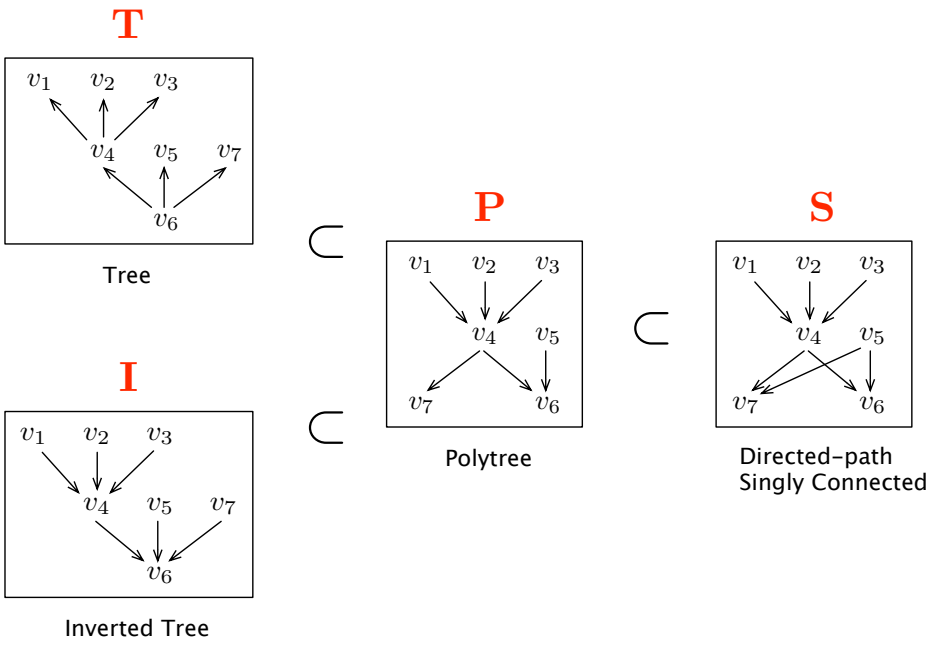


Figure 3.1: Examples of causal graph topologies, along with the inclusion relations between the induced fragments of UB

and Bäckström and Klein (1991), more recent works have studied the impact of posing structural and mixed structural/syntactic restrictions on the action sets. In the scope of UB, most of these works relate the complexity of planning to the topological properties of the problem's *causal graph* structure.

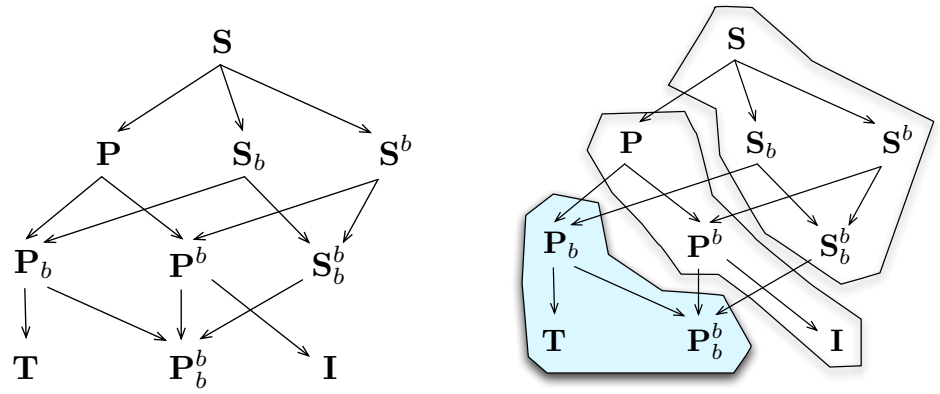
Long before they were used for complexity analysis, causal graphs were (sometimes indirectly) considered in the scope of hierarchically decomposing planning tasks (Newell & Simon, 1963; Sacerdoti, 1974; Knoblock, 1994; Tenenberg, 1991; Bacchus & Yang, 1994). The first result relating the complexity of UB planning to the structure of the causal graph is due to Jonsson and Bäckström (1995, 1998b), who identify a fragment of UB, called **3S**, which has the interesting property of inducing tractable plan existence yet intractable plan generation. One of the key characteristics of **3S** is the acyclicity of the causal graphs. A special case of **3S** was also independently studied by Williams and Nayak (1997) in the scope of incremental planning for more general SAS$^+$ problems.

More recently, Brafman and Domshlak (2003) provide a detailed account of the complexity of finding plans for UB problems with acyclic causal graphs. These results are closely related to the problems examined in this work, and thus we survey them in greater detail. For ease of presentation, we introduce here certain notations that will be used in much of what follows.

- For each node $v \in CG(\Pi)$, by $\mathsf{In}(v)$ and $\mathsf{Out}(v)$ we denote the in- and out-degrees of v, respectively, and $\mathsf{In}(CG(\Pi))/\mathsf{Out}(CG(\Pi))$ stand for the maximal in-degree/out-degree of the $CG(\Pi)$ nodes.

- Assuming $CG(\Pi)$ is connected[1], we provide a special notation for the following topologies of

[1]If $CG(\Pi)$ consists of a few connected components, then these components identify independent sub-problems of Π that can be treated separately.



(a) (b)

Figure 3.2: Inclusion-based hierarchy and complexity of plan generation for some UB problems with acyclic causal graphs. (a) The hierarchy of STRIPS fragments corresponding to tree, inverted tree, polytree, and directed-path singly connected topologies of the causal graph, and (possibly) $O(1)$ bounds on the causal graph in-degree and/or out-degree. (b) Plan generation is tractable for the fragments in the (bottom-most) shaded region, and NP-complete for all other depicted fragments. The top-most and intermediate (transparent) regions correspond to the results of Brafman and Domshlak (2003) and Jonsson and Giménez (2007), respectively.

acyclic causal graphs, also depicted in Figure 3.1. A causal graph $CG(\Pi)$ is a

F *fork* if $\mathsf{Out}(v) > 0$ for exactly one $v \in V$.

IF *inverted fork* if $\mathsf{In}(v) > 0$ for exactly one $v \in V$.

T *tree* if $\mathsf{In}(CG(\Pi)) \leq 1$, and there exists $v \in V$ such that $\mathsf{In}(v) = 0$.

I *inverted tree* if $\mathsf{Out}(CG(\Pi)) \leq 1$, and there exists $v \in V$ such that $\mathsf{Out}(v) = 0$.

P *polytree* if $CG(\Pi)$ contains no undirected cycles. (For an example of a polytree that is neither tree nor inverted tree see Figure 3.1.)

S *directed-path singly connected* if there is at most one directed path from each node $v \in CG(\Pi)$ to any other node $v' \in CG(\Pi)$. (For an example of a directed-path singly connected DAG see Figure 3.1.)

In what follows, we use **T**, **I**, **P**, and **S** to refer to the corresponding fragments of UB, and we use subscript/superscript b to refer to a fragment induced by the additional constraint of in-degree/out-degree being bounded by a constant. It is not hard to verify that we have $\mathbf{T}, \mathbf{I} \subset \mathbf{P} \subset \mathbf{S}$, with $\mathbf{T} \subset \mathbf{P}_b$ and $\mathbf{I} \subset \mathbf{P}^b$; the complete inclusion hierarchy of these sub-fragments of UB is shown in Figure 3.2a. The fragments **F** and **IF** are introduced here, and referred to later on in a different context.

One of the key properties of cost-optimal plans for the UB problems with directed-path singly connected causal graphs is immediately derivable from Lemma 1 of Brafman and Domshlak (2003), and it is given by Corollary 1 below.

Lemma 1 (Brafman and Domshlak (2003)) *For any solvable* UB *task* Π *with causal graph* $CG(\Pi) \in \mathbf{S}$ *over* n *state variables, any irreducible plan* ρ *for* Π*, and any state variable* v *in* Π*, the number of value changes of* v *along* ρ *is* $\leq n$*, that is,* $|\rho\downarrow_v| \leq n$*.*

Corollary 1 *For any solvable* UB *task* Π *with causal graph* $CG(\Pi) \in \mathbf{S}$ *over* n *state variables, any cost-optimal plan* ρ *for* Π*, and any state variable* v *in* Π*, we have* $|\rho\downarrow_v| \leq n$*.*

Given an SAS$^+$ planning task $\Pi = \langle V, A, I, G, cost\rangle$ and a binary-valued variable $v \in V$, we denote the initial value $I[v]$ of v by b_v, and the opposite value by w_v (short for, **black**/**white**). Using this notation and exploiting Corollary 1, by $\sigma(v)$ we denote the **longest possible sequence of values obtainable by** v **along a cost-optimal plan** ρ, with $|\sigma(v)| = n+1$, b_v occupying all the odd positions of $\sigma(v)$, and w_v occupying all the even positions of $\sigma(v)$. In addition, by $\tau(v)$ we denote a per-value **time-stamping of** $\sigma(v)$

$$\tau(v) = \begin{cases} \mathsf{b}_v^1 \cdot \mathsf{w}_v^1 \cdot \mathsf{b}_v^2 \cdot \mathsf{w}_v^2 \cdots \mathsf{b}_v^{j+1}, & n = 2j, \\ \mathsf{b}_v^1 \cdot \mathsf{w}_v^1 \cdot \mathsf{b}_v^2 \cdot \mathsf{w}_v^2 \cdots \mathsf{w}_v^j, & n = 2j-1, \end{cases} , j \in \mathbb{N}.$$

The sequences $\sigma(v)$ and $\tau(v)$ play an important role in our constructions both by themselves and via their prefixes and suffixes. In general, for any sequence S, by $\trianglerighteq[S]$ and $\trianglelefteq[S]$ we denote the set of all non-empty **prefixes** and **suffixes** of S, respectively. In our context, a prefix $\sigma' \in \trianglerighteq[\sigma(v)]$ is called **goal-valid** if either the goal value $G[v]$ is unspecified, or the last element of σ' equals $G[v]$. The set of all goal-valid prefixes of $\sigma(v)$ is denoted by $\trianglerighteq^*[\sigma(v)] \subseteq \trianglerighteq[\sigma(v)]$. The notion of goal-valid prefixes is also similarly specified for $\tau(v)$.

The key tractability result of Brafman and Domshlak (2003) corresponds to a polynomial time plan generation procedure for $\mathbf{P}_b$, that is, for UB problems inducing polytree causal graphs with all nodes having $O(1)$-bounded indegree. In addition, Brafman and Domshlak show that plan generation is NP-complete for the fragment $\mathbf{S}$, and we note that their proof of this claim can be easily modified to hold for $\mathbf{S}_b^b$. These results of tractability and hardness (as well as their immediate implications) are depicted in Figure 3.2b by the shaded bottom-most and the transparent top-most free-shaped regions. The empty free-shaped region in between corresponds to the gap left by Brafman and Domshlak (2003). This gap has been recently closed by Jonsson and Giménez (2007), who prove NP-completeness of plan generation for $\mathbf{P}$. We note that Jonsson and Giménez's proof actually carries over to the $\mathbf{I}$ fragment as well, and so the gap left by Brafman and Domshlak (2003) is now entirely closed.

3.1 Binary Variable Domains

The complexity results of Brafman and Domshlak (2003) and those of Jonsson and Giménez (2007) correspond to satisficing planning, and do not distinguish between the plans on the basis of their quality. By contrast, here we study the complexity of *optimal plan generation* for UB, focusing on (probably the most canonical) *cost-optimal* (also known as sequentially-optimal) planning. Cost-optimal planning corresponds to the task of finding a plan $\rho \in A^*$ for Π that minimizes $cost(\rho) = \sum_{a\in\rho} cost(a)$. We provide novel tractability results for cost-optimal planning for UB, and draw a dividing line between the tractable and intractable such problems.

In the rest of this section our goal is to adequately describe our results, while most formal definitions, constructions, and proofs underlying these results are relegated to appendix A.

3.1.1 Cost-optimal planning for $\mathbf{P}_b$

Following Brafman and Domshlak (2003), here we relate the complexity of (cost-optimal) UB planning to the topology of the causal graph. To this end, we consider the structural hierarchy depicted in Figure 3.2a. We begin by considering cost-optimal planning for $\mathbf{P}_b$—it is apparent from Figure 3.2b that this is the most expressive fragment of the hierarchy that is still a candidate for tractable cost-optimal planning. We prove that cost-optimal planning for $\mathbf{P}_b$ is tractable and show that the complexity map of cost-optimal planning for the UB fragments in Figure 3.2a is *identical* to that for satisficing planning (that is, Figure 3.2b).

Our algorithm for the $\mathbf{P}_b$ fragment is based on compiling a given planning task Π in $\mathbf{P}_b$ into a *constraint optimization problem* $\mathsf{COP}_\Pi = (\mathcal{X}, \mathcal{F})$ over variables $\mathcal{X}$, functional components $\mathcal{F}$, and the global objective $\min \sum_{\varphi \in \mathcal{F}} \varphi(\mathcal{X})$ such that

(I) COP_Π can be constructed in time polynomial in the description size of Π;

(II) the tree-width of the cost network of COP_Π is bounded by a constant, and the optimal tree-decomposition of the network is given by the compilation process;

(III) if Π is unsolvable then all the assignments to $\mathcal{X}$ evaluate the objective function to ∞, and otherwise, the optimum of the global objective is obtained on and only on the assignments to $\mathcal{X}$ that correspond to cost-optimal plans for Π;

(IV) given an optimal solution to COP_Π, the corresponding cost-optimal plan for Π can be reconstructed from the former in polynomial time.

Having such a compilation scheme, we then solve COP_Π using the standard, poly-time algorithm for constraint optimization over trees (Dechter, 2003), and find an optimal solution for Π. The compilation is based on a certain property of the cost-optimal plans for $\mathbf{P}_b$ that allows for conveniently bounding the number of times each state variable changes its value along such an optimal plan. Given this property of $\mathbf{P}_b$, each state variable v is compiled into a single COP variable x_v, and the domain of that COP variable corresponds to all possible sequences of value changes that v may undergo along a cost-optimal plan. The functional components $\mathcal{F}$ are then defined, one for each COP variable x_v, and the scope of such a function captures the "family" of the original state variable v in the causal graph, that is, v itself and its immediate predecessors in $CG(\Pi)$. (See Appendix A for the formal definition of the COP, as well as the correctness and complexity claims.) Figure 3.3a depicts a causal graph of a $\mathbf{P}$ task Π, with the family of the state variable v_4 depicted by the shaded region, and Figure 3.3b shows the cost network induced by compiling Π as a $\mathbf{P}_b$ task, with the dashed line surrounding the scope of the functional component induced by the family of v_4. It is not hard to verify that such a cost network induces a tree over variable-families cliques, and for a $\mathbf{P}_b$ task, the size of each such clique is bounded by a constant. Hence, the tree-width of the cost-network is bounded by a constant as well.

3.1.2 Cost-optimal planning for $\mathbf{P}(1)$

Relaxing the indegree bound brings us to the $\mathbf{P}$ fragment of UB, which is NP-hard even for satisficing planning (Jonsson & Giménez, 2007). Thus, some additional constraint should be introduced.

Causal graphs and k-dependence

While causal graphs provide important information about the structure of planning problems, a closer look at their definition reveals that some accessible information is actually hidden by this structure. To start with an example, let us consider the multi-valued encoding of Logistics-style problems (Helmert, 2006). In these problems, each variable representing the location of a package has as its parents in the causal graph *all* the variables representing alternative transportation means (trucks, planes, etc.), and yet, each individual action affecting the location of a package is preconditioned by at most *one* such parent variable. (You cannot load/unload a package into/from more than one vehicle.) So, even if the in-degree of the causal graph is proportional to some problem domain parameters, the number of variables that determine applicability of each action may still be bounded by a constant.

In other words, while the causal graph provides an aggregate view of the independence relationships between the problem variables, the individual dependencies of the problem actions on the non-changing variables are suppressed by this view. To exploit these dependencies, we propose a (tangential to the causal graph's topology) classification of UB problems and study the connection between this classification and the computational tractability of both general and cost-optimal plan generation for UB.

Definition 8 *For any $k \in \mathbb{Z}^*$, and any* SAS$^+$ *task Π over actions A, we say that Π is* **k-dependent** *if it satisfies* $|\mathcal{V}(\mathsf{pre}(a)) \setminus \mathcal{V}(\mathsf{eff}(a))| \leq k$ *for all actions $a \in A$*

In other words, an SAS$^+$ planning task is k-dependent if no action in its action set depends on more than k non-changing variables. Putting our two classifications of problems together, for any structural fragment $\mathbf{G}$ of UB (such as, e.g., those in Figure 3.2), and any $k \in \mathbb{Z}^*$, by $\mathbf{G}(k)$ we denote the set of all k-dependent problems within $\mathbf{G}$.

Meeting Polytrees and 1-dependence

Recall that the fragment $\mathbf{P}$ of UB is NP-hard even for satisficing planning (Jonsson & Giménez, 2007). Our main result here is positive—at least for the most extreme (yet, says the Logistics example above, not unrealistic) setting of $k = 1$, satisfying k-dependence does bring us to an island of tractability $\mathbf{P}(1)$.

Similarly to our treatment of $\mathbf{P}_b$, our algorithm for $\mathbf{P}(1)$ exploits the idea of compiling a planning task Π into a tractable constraint optimization problem COP_Π. However, the planning-to-COP compilation scheme here is very different from that devised for $\mathbf{P}_b$. In fact, this difference is unavoidable since our construction for $\mathbf{P}_b$ relies heavily on the assumption that $\mathsf{ln}(CG(\Pi)) = O(1)$, and we do not have this luxury in $\mathbf{P}(1)$. Instead, we identify certain properties of the cost-optimal plan sets of the $\mathbf{P}(1)$ tasks, and exploit these properties in devising suitable planning-to-COP compilation schemes.

We begin with considering only $\mathbf{P}(1)$ tasks with uniform-cost actions; the cost of a plan for such a task is proportional to its length [2]. We show that any such solvable task has a cost-optimal plan that makes all the changes of each variable to a certain value using exactly the same (type of) action. And while devising a correct and tractable planning-to-COP compilation scheme is more than a step away from identifying this property of $\mathbf{P}(1)$, the latter is the cornerstone of what follows. This property of $\mathbf{P}(1)$ can be used to uniquely compile each state variable v and *each*

[2] This is probably the origin of the term "sequential optimality."

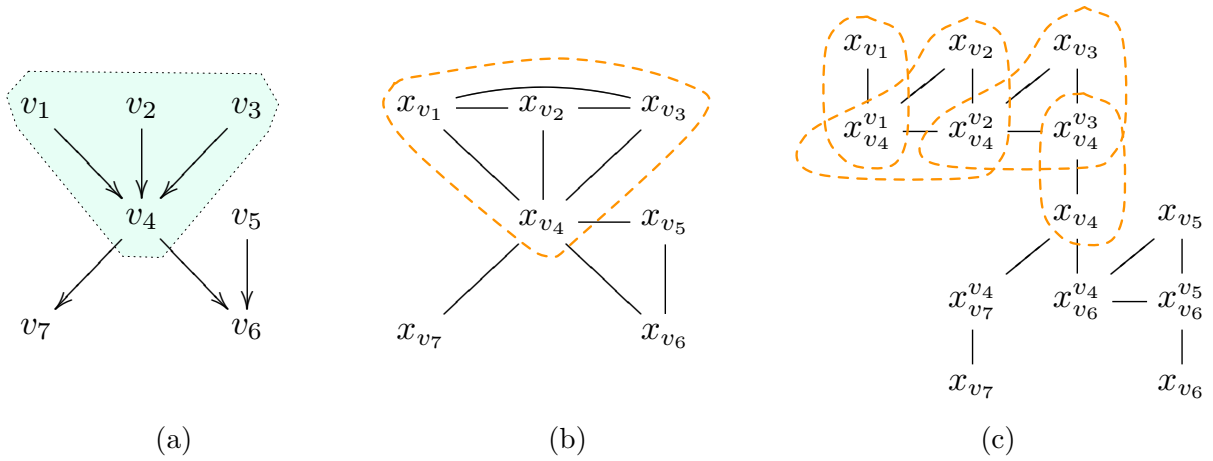


Figure 3.3: Cost networks induced by the planning-to-COP compilation schemes for $\mathbf{P}$. (a) Causal graph of a $\mathbf{P}$ task Π, with the family of the state variable v_4 being depicted by the shaded region. (b) Cost network induced by compiling Π as a $\mathbf{P}_b$ task, with the dashed line surrounding the scope of the functional component induced by the family of v_4. (c) Cost network induced by compiling Π as a $\mathbf{P}(1)$ task, with the dashed lines surrounding the scopes of the four functional components induced by the family of v_4.

edge (v, v') of the task's causal graph into COP variables x_v and $x_{v'}^v$ (see Figure 3.3c). A certain *set* of functional components is then defined for each COP variable x_v. Because the domains of the COP variables and the specification of the functional components are technically complex, we relegate them to Appendix A. It is important, however, to note that the cost networks of such COPs are guaranteed to induce trees over cliques of size ≤ 3, and thus having tree-width bounded by a constant. The reader can get an intuition of where these "cliques of size ≤ 3" are coming from by looking at the example in Figure 3.3c.

Unfortunately, the aforementioned helpful property of the $\mathbf{P}(1)$ tasks with uniform-cost actions does not hold for more general action-cost schemes for $\mathbf{P}(1)$. It turns out, however, that *all* tasks in $\mathbf{P}(1)$ satisfy another property that still allows for devising a general, correct, and tractable planning-to-COP scheme for $\mathbf{P}(1)$. Specifically, we show that any solvable task in $\mathbf{P}(1)$ has a cost-optimal plan that makes all the changes of each variable using at most three types of action. The algorithm resulting from exploiting this property is more complex and more costly than that devised for the $\mathbf{P}(1)$ tasks with uniform-cost actions, yet it is still poly-time. Interestingly, the cost networks of such COPs are topologically identical to those for problems with uniform-cost actions, the difference being in the domains of the COP variables, and in the specification of the functional components.

3.1.3 Drawing the Limits of k-dependence

Having read this far, the reader may rightfully wonder whether $O(1)$-dependence is not a strong enough property to make cost-optimal planning tractable even for some forms of the causal graph that are more complex than polytree. It turns out that that the dividing line between tractable

	$k=1$	$k=2$	$k=3$	$k=\Theta(n)$
$\mathbf{P}_b$	—	—	—	P
$\mathbf{P}(k)$	P			NPC
$\mathbf{S}_b^b$	NPC	—	—	NPC

(a)

	$k=1$	$k=2$	$k=3$	$k=\Theta(n)$
$\mathbf{P}_b$	—	—	—	P
$\mathbf{P}(k)$	P			NPC
$\mathbf{S}_b^b$		NPC	—	NPC

(b)

Figure 3.4: Complexity of (a) cost-optimal and (b) satisficing plan generation for fragments of UB. The "—" mark indicates that the complexity is implied by other results in the row. All other shaded and regular cells correspond to the results obtained in this work and in the past, respectively. Empty cells correspond to open questions. Note that, with the exception of $\mathbf{S}(1)$ fragment, the complexity of both cost-optimal and satisficing planning is either unknown or the same.

and intractable cases is much finer. Figure 3.4 summarizes our current understanding of the time complexity of both cost-optimal and satisficing plan generation for the **P** and **S** fragments of UB. In this section we show that even satisficing planning with directed-path singly connected, bounded in- and out-degree causal graphs is hard under 2-dependence, and that cost-optimal planning for this structural fragment of UB is hard even for 1-dependent such tasks. Note that the complexity of (both cost-optimal and satisficing) plan generation for $\mathbf{P}(k)$ for $k = O(1)$, $k \geq 2$ remains an interesting open problem. An additional open question is the complexity of satisficing plan generation for $\mathbf{S}(1)$. The results for the **S** fragments are obtained as follows.

- In Theorem 1 we show that cost-optimal planning is already hard for the $\mathbf{S}_b^b(1)$ class, that is, the class of 1-dependent UB tasks inducing directed-path singly connected causal graphs with both in- and out-degrees being bounded by a constant. This result further emphasizes the connection between the undirected cycles in the causal graph and the complexity of various planning tasks, which was first discussed by Brafman and Domshlak (2003).

- In Theorem 2 we show that even non-optimal planning is hard for the $\mathbf{S}_b^b(2)$ class. This result suggests that 1-dependence is a rather special case of k-dependence in terms of the connection to computational tractability. However, given the (still) empty entries in Figures 3.4a and 3.4b, further analysis of the "criticality" of 1-dependence is needed.

Theorem 1 *Cost-optimal planning for* $\mathbf{S}_b^b(1)$ *is NP-complete.*

Proof: The membership in NP is by Theorem 2 of Brafman and Domshlak (2003). The proof of NP-hardness is by a polynomial reduction from the well-known VERTEX COVER problem (Garey & Johnson, 1978): given an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, find a minimal-size subset $\mathbf{V}'$ of $\mathbf{V}$ such that each edge in $\mathbf{E}$ has at least one of its two end-nodes in $\mathbf{V}'$. Given an undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, let the planning task $\Pi_{\mathbf{G}} = \langle V_{\mathbf{G}}, A_{\mathbf{G}}, I_{\mathbf{G}}, G_{\mathbf{G}}, cost \rangle$ be defined as follows.

- $V_{\mathbf{G}} = \{v_1, \ldots, v_{|\mathbf{V}|}, u_1, \ldots, u_{|\mathbf{E}|}\}$, and, for all v_i, u_j, $\mathcal{D}(v_i) = \mathcal{D}(u_j) = \{T, F\}$,
- $I_{\mathbf{G}} = \{v_i = F \mid v_i \in V_{\mathbf{G}}\} \bigcup \{u_i = F \mid u_i \in V_{\mathbf{G}}\}$,
- $G_{\mathbf{G}} = \{u_i = T \mid u_i \in V_{\mathbf{G}}\}$,

- Actions $A_{\mathbf{G}} = A_{\mathbf{V}} \cup A_{\mathbf{E}}$, where $A_{\mathbf{V}} = \{a_{v_1}, \ldots, a_{v_{|\mathbf{V}|}}\}$ with

$$\mathsf{pre}(a_{v_i}) = \{v_i = F\}, \;\; \mathsf{eff}(a_{v_i}) = \{v_i = T\}, \;\; cost(a_{v_i}) = 1 \tag{3.1}$$

and $A_{\mathbf{E}} = \{a_{u_1}, a'_{u_1}, \ldots, a_{u_{|\mathbf{E}|}}, a'_{u_{|\mathbf{E}|}}\}$ with

$$\begin{aligned} \mathsf{pre}(a_{u_i}) &= \{u_i = F, v_{i_1} = T\}, \\ \mathsf{pre}(a'_{u_i}) &= \{u_i = F, v_{i_2} = T\}, \\ \mathsf{eff}(a_{u_i}) = \mathsf{eff}(a'_{u_i}) &= \{u_i = T\}, \\ cost(a_{u_i}) = cost(a'_{u_i}) &= 0 \end{aligned} \tag{3.2}$$

where the variables v_{i_1}, v_{i_2} correspond to the endpoints of the edge corresponding to the variable u_i.

Given this construction of $\Pi_{\mathbf{G}}$, it is easy to see that (i) any plan ρ for $\Pi_{\mathbf{G}}$ provides us with a vertex cover $\mathbf{V}_\rho$ for $\mathbf{G}$ such that $|\mathbf{V}_\rho| = cost(\rho)$ and vice versa, and thus (ii) cost-optimal plans for $\Pi_{\mathbf{G}}$ (and only such plans for $\Pi_{\mathbf{G}}$) provide us with minimal vertex covers for $\mathbf{G}$. The topology of the causal graph of $\Pi_{\mathbf{G}}$ is as required, and 1-dependence of $\Pi_{\mathbf{G}}$ is immediate from Eqs. 3.1-3.2. This finalizes the proof of NP-completeness of cost-optimal planning for $\mathbf{S}^b_b(1)$. ∎

Theorem 2 *Satisficing planning for* $\mathbf{S}^b_b(2)$ *is NP-complete.*

Proof: The proof is, in essence, the proof for Theorem 2 of Brafman and Domshlak (2003). The polynomial reduction there is from 3-SAT to planning for $\mathbf{S}$. Observing that 3-SAT remains hard even if no variable participates in more than five clauses of the formula (Garey & Johnson, 1978), and that the reduction of Brafman and Domshlak from *such* 3-SAT formulas is effectively to satisficing planning for $\mathbf{S}^b_b(2)$, we prove our claim. ∎

3.1.4 Towards Practically Efficient Special Cases

The polytree-k-indegree algorithm for $\mathbf{P}_b$ is polynomial, but is rather involved and its complexity is exponential in $\mathsf{In}(CG(\Pi))$. It is quite possible that more efficient algorithms for $\mathbf{P}_b$, or some of its fragments can be devised. A simple algorithm for $\mathbf{T} \subset \mathbf{P}_b$ tasks has indeed already appeared in the literature in a different context, but when or even if it provides cost-optimal solutions was never examined. This is the TreeDT algorithm for preferential reasoning with tree-structured CP-nets (Boutilier, Brafman, Domshlak, Hoos, & Poole, 2004), and it turns out that its straightforward adaptation for $\mathbf{T}$ planning tasks always provides cost-optimal solutions for $\mathbf{T}$ tasks with *uniform-cost actions*. The algorithm is depicted in Figure 3.5, and it is not hard to verify that its time complexity is linear in the length of the generated plan ρ—all it does is iteratively "remove" the parts of the task that can be safely ignored in later steps, and then apply a value-changing action on a lowest (in the causal graph) variable for which such an action exists.

Theorem 3 *Given a* $\mathbf{T}$ *task* Π *with uniform-cost actions over* n *state variables,*

(I) the algorithm tree-uniform-cost *finds a plan if and only if* Π *is solvable,*

(II) if the algorithm tree-uniform-cost *finds a plan for* Π*, then this plan is cost-optimal, and*

(III) the time complexity of tree-uniform-cost *is* $\Theta(n^2)$.

procedure tree-uniform-cost($\Pi = \langle V, A, I, G, cost \rangle$)
takes a planning task $\Pi \in \mathbf{T}$ with uniform-cost actions A
returns a cost-optimal plan for Π if Π is solvable, and fails otherwise
$\rho = \langle\rangle$, $s := I$, $V' := V$
loop
$V' :=$ remove-solved-leafs(s, V')
if $V' = \emptyset$ **return** ρ
else
find $v \in V', a \in A_v$ such that $a \in A(s)$ and
$\forall u \in \mathsf{Desc}(v, V') : \ A_u \cap A(s) = \emptyset$
if not found **return** failure
$\rho := \rho \cdot \langle a \rangle$, $s := s[\![a]\!]$

Figure 3.5: A simple algorithm for cost-optimal planning for $\mathbf{T}$ tasks with uniform-cost actions. The notation $\mathsf{Desc}(v, V')$ stands for the subset of V' containing the descendants of v in $CG(\Pi)$, and $A(s)$ stands for the set of all actions applicable in the state s. The function remove-solved-leafs(s, V') iteratively removes the leaf variables that are in their goal value (if defined) until no such variable is found.

Proof: Without loss of generality, in what follows we assume that the actions of Π are all unit-cost, that is, for each plan ρ for Π, $cost(\rho) = |\rho|$.

(I) Straightforward by reusing *as is* the proof of Theorem 11 of Boutilier et al. (2004).

(II) Assume to the contrary that the plan ρ provided by tree-uniform-cost is not optimal, that is, there exists a plan ρ' such that $|\rho'| < |\rho|$. In particular, this implies the existence of a variable v such that $|\rho'{\downarrow_v}| < |\rho{\downarrow_v}|$. The semantics of planning imply that

$$|\rho'{\downarrow_v}| \leq |\rho{\downarrow_v}| - (\epsilon_v + 1) \tag{3.3}$$

where $\epsilon_v = 1$ if $G[v]$ is specified, and 0 otherwise. Likewise, since the causal graph $CG(\Pi)$ forms a directed tree, there exists a variable v satisfying Eq. 3.3 such that, for all the descendants u of v in $CG(\Pi)$ holds:

$$|\rho'{\downarrow_u}| \geq |\rho{\downarrow_u}| \tag{3.4}$$

Let $Ch(v)$ be the set of all the immediate descendants of v in $CG(\Pi)$. By the construction of tree-uniform-cost, we have that:

1. If $Ch(v) = \emptyset$, then $|\rho{\downarrow_v}| \leq \epsilon_v$, and this contradicts Eq. 3.3 as $|\rho'{\downarrow_v}|$ is a non-negative quantity by definition.

2. Otherwise, if $Ch(v) \neq \emptyset$, then, by the construction of tree-uniform-cost, there exists $u \in Ch(v)$ such that changing its value $|\rho{\downarrow_u}|$ times *requires* changing the value of v at least $|\rho{\downarrow_v}| - \epsilon_v$ times. In other words, there is no action sequence ϱ applicable in I such that $|\varrho{\downarrow_u}| \geq |\rho{\downarrow_u}|$ while $|\varrho{\downarrow_v}| < |\rho{\downarrow_v}| - \epsilon_v$. However, from Eq. 3.4 we have $|\rho'{\downarrow_u}| \geq |\rho{\downarrow_u}|$, and thus $|\rho'{\downarrow_v}|$ has to be at least $|\rho{\downarrow_v}| - \epsilon_v$. This, however, contradicts Eq. 3.3.

Hence, we have proved that $|\rho'{\downarrow_v}| \geq |\rho{\downarrow_v}|$, contradicting our assumption that $|\rho'| < |\rho|$.

(III) Implied by Theorems 12 and 13 of Boutilier et al. (2004). ∎

The requirement in Theorem 3 for all actions to have the same cost is essential. The example below shows that in the more general case the algorithm tree-uniform-cost is no longer cost-optimal. Consider $\Pi = \langle V, A, I, G, cost\rangle \in \mathbf{T}$ with $V = \{v, u\}$, $I = \{\mathsf{b}_v, \mathsf{b}_u\}$, $G = \{\mathsf{b}_v, \mathsf{w}_u\}$, and $A = \{a_1, a_2, a_3, a_4\}$ with

$$\begin{aligned}
&\mathsf{eff}(a_1) = \{\mathsf{w}_v\}, \mathsf{pre}(a_1) = \{\mathsf{b}_v\}\\
&\mathsf{eff}(a_2) = \{\mathsf{b}_v\},\ \mathsf{pre}(a_2) = \{\mathsf{w}_v\}\\
&\mathsf{eff}(a_3) = \{\mathsf{w}_u\}, \mathsf{pre}(a_3) = \{\mathsf{b}_u, \mathsf{w}_v\}\\
&\mathsf{eff}(a_4) = \{\mathsf{w}_u\}, \mathsf{pre}(a_4) = \{\mathsf{b}_u, \mathsf{b}_v\}\\
&\quad cost(a_1) = cost(a_2) = cost(a_3) = 1\\
&\quad cost(a_4) = 4.
\end{aligned}$$

On this task, the tree-uniform-cost algorithm returns $\rho = \langle a_4\rangle$ with $cost(\rho) = 4$, while the optimal plan is $\rho' = \langle a_1, a_3, a_2\rangle$ with $cost(\rho') = 3$.

3.2 General Variable Domains

Investigating the complexity of SAS$^+$ fragments beyond UB is not of theoretical interest alone. Relaxing the unary effectness property leads to complex-structured causal graphs, almost immedietly taking us beyond the limits of tractability. Unfortunately, relaxing domain bounds altogether also can harm the tractability of even the simplest causal graph structures. Domshlak and Dinitz (2001) show that even satisficing planning for SAS$^+$ fragments with fork and inverted fork causal graphs is NP-complete. In fact, recent results by Chen and Giménez (2008) show that planning for any SAS$^+$ fragment characterized by any non-trivial connected causal graph is NP-hard. Moreover, even if the domain-transition graphs of all the state variables are strongly connected (as in our example), optimal planning for forks and inverted forks remain NP-hard (see Helmert (2003) and (2004) for the respective results).

While the hardness of optimal planning for problems with fork and inverted fork causal graphs casts a shadow on the relevance of fork-decompositions, a closer look at the proofs of the corresponding hardness results of Domshlak and Dinitz (2001) and Helmert (2003, 2004) reveals that these proofs rely on root variables having large domains. It turns out that this is not coincidental, and Theorems 4 and 5 below characterize some substantial islands of tractability within these structural fragments of SAS$^+$.

3.2.1 Cost-optimal planning for F

In order to investigate the complexity of tasks with fork-structured causal graphs and domain-bounded variables, one should distinguish between the bounds on the domains of the leaf variables and the bound on root domain. Here we present the complexity results for the following two cases: (i) binary root domain and unbounded domains for the leaf variables, and (ii) constant bound on all task variables. Together with the NP-Completeness results for the unbounded root domain case,

the only remaining open question is the constant bounded root domain and unbounded domains for the leaf variables. The tractability proofs for the aforementioned two cases are given in Theorem 4.

Theorem 4 (Tractable Forks) *Given a planning task $\Pi = \langle V, A, I, G, cost\rangle$ with a fork causal graph rooted at r, if (i) $|\mathcal{D}(r)| = 2$, or (ii) for all $v \in V$, $|\mathcal{D}(v)| = O(1)$, the time complexity of the cost-optimal planning for Π is polynomial in $||\Pi||$.*

Proof: Observe that, for any planning task Π as in the theorem, the fork structure of the causal graph $CG(\Pi)$ implies that all the actions in Π are unary-effect, and each leaf variable $v \in \mathsf{succ}(r)$ preconditions only the actions affecting v itself. We prove the two cases separately.

First, for $|\mathcal{D}(r)| = 2$, the algorithm below is based on the following three properties satisfied by the optimal plans ρ for Π.

(i) For any leaf variable $v \in \mathsf{succ}(r)$, the path $\rho{\downarrow_v}$ from $I[v]$ to $G[v]$ induced by ρ in $DTG(v, \Pi)$ is either cycle-free or contains only zero-cost cycles. This is the case because otherwise all the nonzero-cost cycles can be eliminated from $\rho{\downarrow_v}$ while preserving its validity, violating the assumed optimality of ρ. Without loss of generality, in what follows we assume that this path $\rho{\downarrow_v}$ in $DTG(v, \Pi)$ is cycle-free; in the case of fork causal graphs, we can always select an optimal ρ that satisfies this requirement for all $v \in \mathsf{succ}(r)$. Thus, we have $|\rho{\downarrow_v}| \leq |\mathcal{D}(v)| - 1$.

(ii) *Having fixed* a sequence of value changes of r, the fork's leaves become mutually independent; that is, our ability to change the value of one does not affect our ability to change the value of all the others.

(iii) *Because r is binary-valued*, if $v \in V \setminus \{r\}$ is the "most demanding" leaf variable in terms of the number of value changes required from r by the action preconditions along $\rho{\downarrow_v}$, then these are the only value changes of r along ρ, except for, possibly, a final value change to $G[r]$. Thus, in particular, we have $|\rho{\downarrow_r}| \leq \max_{v\in\mathsf{succ}(r)} |\mathcal{D}(v)|$.

We begin with introducing some auxiliary notations. With $|\mathcal{D}(r)| = 2$, without loss of generality let $\mathcal{D}(r) = \{0, 1\}$ with $I[r] = 0$. Let $\sigma(r)$ be an alternating 0/1 sequence starting with 0, and having 0 in all odd and 1 in all even positions. This sequence $\sigma(r)$ is such that $|\sigma(r)| = 1$ if no action in A can change r's value to 1, $|\sigma(r)| = 2$ if some action can change r's value to 1 but no action can then restore it to value 0, and otherwise, $|\sigma(r)| = 1 + \max_{v\in\mathsf{succ}(r)} |\mathcal{D}(v)|$. Let $\trianglerighteq^*[\sigma(r)]$ be the set of all nonempty prefixes of $\sigma(r)$ if $G[r]$ is unspecified; otherwise, let it be the set of all nonempty prefixes of $\sigma(r)$ ending with $G[r]$. Note that if $\trianglerighteq^*[\sigma(r)] = \emptyset$, then the problem is trivially unsolvable; in what follows we assume this is not the case. For each $v \in \mathsf{succ}(r)$, let DTG^0_v and DTG^1_v be the subgraphs of the domain transition graphs $DTG(v, \Pi)$, obtained by removing from $DTG(v, \Pi)$ all the arcs labeled with $r = 1$ and $r = 0$, respectively.

The algorithm below incrementally constructs a set $\mathcal{R}$ of valid plans for Π, starting with $\mathcal{R} = \emptyset$.

(1) For each $v \in \mathsf{succ}(r)$, and each pair of v's values $x, y \in \mathcal{D}(v)$, compute the shortest (that is, cost-minimal) paths $\pi^0_v(x, y)$ and $\pi^1_v(x, y)$ from x to y in DTG^0_v and DTG^1_v, respectively. For some pairs of values x, y, one or even both these paths may, of course, not exist.

(2) For each sequence $\sigma \in \trianglerighteq^*[\sigma(r)]$, and each $v \in \mathsf{succ}(r)$, construct a layered digraph $\mathcal{L}_v(\sigma)$ with $|\sigma|+1$ node layers $L_0, \ldots, L_{|\sigma|}$, where L_0 consists of only $I[v]$, and for $1 \leq i \leq |\sigma|$, L_i consists of all nodes $y \in \mathcal{D}(v)$ for which a path $\pi^{\sigma[i]}_v(x, y)$ from some node $x \in L_{i-1}$ has been constructed in step (1). For each $x \in L_{i-1}, y \in L_i$, $\mathcal{L}_v(\sigma)$ contains an arc (x, y) weighted with $cost(\pi^{\sigma[i]}_v(x, y))$.

(3) For each $\sigma \in \trianglerighteq^*[\sigma(r)]$, let $k = |\sigma|$. A candidate plan ρ_σ for Π is constructed as follows.

   (a) For each $v \in \mathsf{succ}(r)$, find a cost-minimal path from $I[v]$ to $G[v]$ in $\mathcal{L}_v(\sigma)$. If no such path exists, then proceed with the next prefix in $\trianglerighteq^*[\sigma(r)]$. Otherwise, note that the i-th edge on this path (taking us from some $x \in L_{i-1}$ to some $y \in L_i$) corresponds to the cost-minimal path $\pi_v^{\sigma[i]}(x, y)$ from x to y. Let us denote this path from x to y by S_v^i.

   (b) Set $\mathcal{R} = \mathcal{R} \cup \{\rho_\sigma\}$, where $\rho_\sigma = S^1 \cdot a_{\sigma[2]} \cdot S^2 \cdot \ldots \cdot a_{\sigma[k]} \cdot S^k$, each sequence S^i is obtained by an *arbitrary* merge of the sequences $\{S_v^i\}_{v \in \mathsf{succ}(r)}$, and a_ϑ is the cheapest action changing the value of r to value ϑ.

(4) If $\mathcal{R} = \emptyset$, then fail, otherwise return $\rho = \mathrm{argmin}_{\rho_\sigma \in \mathcal{R}}\, cost(\rho_\sigma)$.

It is straightforward to verify that the complexity of the above procedure is polynomial in the description size of Π. To prove correctness, we show that the procedure returns a plan for any solvable task Π, and that the returned plan ρ' satisfies $cost(\rho') \leq cost(\rho)$ for any optimal plan ρ for Π.

Given a solvable task Π, let ρ be an optimal plan for Π with all $\rho\downarrow_v$ for the leaf variables v being cycle-free. Let $\rho\downarrow_r = \langle a_2 \ldots, a_k \rangle$; the numbering of actions along $\rho\downarrow_r$ starts with a_2 to simplify indexing later on. For each $v \in \mathsf{succ}(r)$, the actions of $\rho\downarrow_r$ divide $\rho\downarrow_v$ into subsequences of v-changing actions $\rho\downarrow_v = \rho_v^1 \cdot \ldots \cdot \rho_v^k$, separated by the value changes required from r. That is, for each $1 \leq i \leq k$, all actions in ρ_v^i are preconditioned by the same value of r, if any, and if two actions $a \in \rho_v^i$ and $a' \in \rho_v^{i+1}$ are preconditioned by r, then $\mathsf{pre}(a)[r] \neq \mathsf{pre}(a')[r]$. Let $\sigma \in \trianglerighteq^*[\sigma(r)]$ be a value sequence such that $|\sigma| = k = |\rho\downarrow_r| + 1$. For each $v \in \mathsf{succ}(r)$, $\rho\downarrow_v$ is a path from $I[v]$ to $G[v]$ in $\mathcal{L}_v(\sigma)$, and therefore some ρ_σ is added into $\mathcal{R}$ by the algorithm, meaning that the algorithm finds a solution. Now, if $\rho_\sigma \in \mathcal{R}$, then, for each $v \in \mathsf{succ}(r)$, let $S_v^1 \cdot S_v^2 \cdot \ldots \cdot S_v^k$ be a cost-minimal path from $I[v]$ to $G[v]$ in $\mathcal{L}_v(\sigma)$ such that S_v^i is the sequence of actions changing the value of v and preconditioned either by $r = 0$ or nothing for odd i, and by $r = 1$ or nothing for even i. Thus,

$$cost(S_v^1 \cdot S_v^2 \cdot \ldots \cdot S_v^k) = \sum_{i=1}^{k} cost(S_v^i) \leq cost(\rho\downarrow_v).$$

Because sequence S^i is obtained by an arbitrary merge of the sequences $\{S_v^i\}_{v \in \mathsf{succ}(r)}$, and a_ϑ is the cheapest action changing the value of r to ϑ, then $\rho_\sigma = S^1 \cdot a_{\sigma[2]} \cdot S^2 \cdot \ldots \cdot a_{\sigma[k]} \cdot S^k$ is an applicable sequence of actions that achieves the goal values for each $v \in \mathsf{succ}(r)$ as well as for r, and

$$\begin{aligned} cost(\rho_\sigma) = cost(S^1 \cdot a_{\sigma[2]} \cdot S^2 \cdot \ldots \cdot a_{\sigma[k]} \cdot S^k) = \sum_{i=2}^{k} cost(a_{\sigma[i]}) + \sum_{i=1}^{k} cost(S^i) \leq \\ \leq cost(\rho\downarrow_r) + \sum_{v \in \mathsf{succ}(r)} cost(\rho\downarrow_v) = cost(\rho). \end{aligned}$$

Hence, if Π is solvable, then the algorithm returns a plan for Π, and this plan must be optimal. Finally, if Π is not solvable, then $\mathcal{R}$ necessarily remains empty, and thus the algorithm fails.

Now we proceed with the second case of the theorem. If $|\mathcal{D}(v)| = O(1)$ for all $v \in V$, let $d = \max_{v \in V} |\mathcal{D}(v)|$. Assuming a unified labeling of the values in each $\mathcal{D}(v)$ as $\{1, 2, \ldots, |\mathcal{D}(v)|\}$, note that (i) there are at most d^d different sequences of domain values of size $\leq d$, (ii) any plan ρ for Π changes the value of each $u \in \mathsf{succ}(r)$ according to one such sequence, and (iii) each value

change of u along ρ is (possibly) prevailed by some value of r. Hence, the length of a sequence of r's values required to support the value changes of $\mathsf{succ}(r)$ along a cost-optimal plan for Π is $\Theta(d^{d+1})$. Since each value of r is reachable from another value of r in $< d$ steps, the total number of value changes of r along a cost-optimal plan for Π is $\Theta(d^{d+2})$.

Given that, we can now generate all possible paths of size $\leq d^{d+2}$ from $I[r]$ to $G[r]$ in $DTG(r, \Pi)$ in time $\Theta(d^{d^{d+2}})$, and, for each such path, and each $u \in \mathsf{succ}(r)$, find a minimal-cost path from $I[u]$ to $G[u]$ in $DTG(u, \Pi)$ that can be supported by the given path for r. This can be done by going over all cycle-free paths in $DTG(u, \Pi)$ from $I[u]$ to $G[u]$, and checking whether a series of values of r that supports the corresponding actions is a subsequence of the path for r. There are $\Theta(d^d)$ such paths in $DTG(u, \Pi)$, and the test per path can be done in time $\Theta(d^{d+2})$. Thus, for each considered path for r in $DTG(r, \Pi)$, the respective minimal-cost path for u in $DTG(u, \Pi)$ can be found in time $\Theta(d^{2d+2})$, and therefore a cost-optimal plan for Π can be found in time $\Theta(d^{2d+2} \cdot d^{d^{d+2}}) = \Theta(d^{d^{d+2}+2d+2}) = O(1)$.[3] ∎

3.2.2 Cost-optimal planning for IF

Theorem 5 (Tractable Inverted Forks) *Given a planning task $\Pi = \langle V, A, I, G, cost \rangle$ with an inverted fork causal graph with sink $r \in V$, if $|\mathcal{D}(r)| = O(1)$, the time complexity of the cost-optimal planning for Π is polynomial in $||\Pi||$.*

Proof: Let $|\mathcal{D}(r)| = d$. Observe that the inverted-fork structure of the causal graph $CG(\Pi)$ implies that all the actions in Π are unary-effect, and that the sink r preconditions only the actions affecting r itself. Hence, in what follows we assume that $G[r]$ is specified; otherwise Π breaks down to a set of trivial planning problems over a single variable each. Likewise, from the above properties of Π it follows that, if ρ is an optimal plan for Π, then the path $\rho{\downarrow_r}$ from $I[r]$ to $G[r]$ induced by ρ in $DTG(r, \Pi)$ is either cycle-free or contains only zero-cost cycles. The latter can be safely eliminated from ρ, and thus we can assume that $\rho{\downarrow_r}$ is cycle-free. Given that, a simple algorithm that finds a cost-optimal plan for Π in time $\Theta(||\Pi||^d + ||\Pi||^3)$ is as follows.

(1) Create all $\Theta(|A_r|^{d-1})$ cycle-free paths from $I[r]$ to $G[r]$ in $DTG(r, \Pi)$.

(2) For each variable $v \in \mathsf{pred}(r)$, and each pair of v's values $x, y \in \mathcal{D}(v)$, compute the cost-minimal path from x to y in $DTG(v, \Pi)$. The whole set of such cost-minimal paths can be computed using $\Theta(d|V|)$ applications of the Floyd-Warshall algorithm on the domain transition graphs of the sink's parents $\mathsf{pred}(r)$.

(3) For each $I[r]$-to-$G[r]$ path in $DTG(r, \Pi)$ generated in step (1), construct a plan for Π based on that path for r, and the shortest paths computed in (2). This simple construction, depicted in Figure 3.6, is possible because the values of each parent variable can be changed independently of the values of all other variables in the inverted fork.

(4) Take the cheapest plan among those constructed in (3). If no plan was constructed in step (3), then Π is unsolvable.

We have already observed that, for each cost-optimal plan ρ, $\rho{\downarrow_r}$ is one of the $I[r]$-to-$G[r]$ paths generated in step (1). For each $v \in \mathsf{pred}(r)$, let S_v denote the sequence of values from $\mathcal{D}(v)$

[3] This constant bound is rather ludicrous, but our construction here is not intended to be complexity-optimal either. Finding more realistic bounds for this concrete problem is definitely of interest.

Given a path $\langle a_1, \ldots, a_m \rangle$ from $I[r]$ to $G[r]$ in $DTG(r, \Pi)$:
$\rho := \langle\rangle$
$a_{m+1} := \langle G[\mathsf{pred}(r)], \emptyset \rangle$
foreach $v \in \mathsf{pred}(r)$ **do** $x_v := I[v]$
for $i := 1$ to $m + 1$ **do**
    foreach $v \in \mathsf{pred}(r)$ **do**
        if $\mathsf{pre}(a_i)[v]$ is specified **and** $\mathsf{pre}(a_i)[v] \neq x_v$ **then**
            if $\mathsf{pre}(a_i)[v]$ is not reachable from x_v in $DTG(v, \Pi)$ **then fail**
            append to ρ the actions induced by some cost-minimal path
                from $\mathsf{pre}(a_i)[v]$ to x_v in $DTG(v, \Pi)$
            $x_v := \mathsf{pre}(a_i)[v]$
    if $i < m + 1$ **then append** to ρ the action a_i
return ρ

Figure 3.6: Detailed outline of step (3) of the planning algorithm for inverted-fork structured task

that is required by the preconditions of the actions along $\rho\downarrow_r$. For each $v \in \mathsf{pred}(r)$, we have $\rho\downarrow_v$ corresponding to a (possibly cyclic) path from $I[v]$ to $G[v]$ in $DTG(v, \Pi)$, traversing the values (= nodes) from S_v in the order required by S_v. In turn, the plan for Π generated in (3) consists of cost-minimal such paths for all $v \in \mathsf{pred}(r)$. Therefore, at least one of the plans generated in (3) must be cost-optimal for Π, and the minimization step (4) will select one of them. ∎

Chapter 4

Implicit Abstractions

Over the years, PDB heuristics have been shown to be very effective in several hard search problems, including cost-optimal planning (Culberson & Schaeffer, 1998; Edelkamp, 2001; Felner et al., 2004; Haslum et al., 2007). The conceptual limitation of these heuristics, however, is that the size of the abstract space and its dimensionality must be fixed.[1] The more recent merge-and-shrink abstractions generalize PDB heuristics to overcome the latter limitation (Helmert et al., 2007). Instead of perfectly reflecting just a few state variables, merge-and-shrink abstractions allow for imperfectly reflecting all variables. As demonstrated by the formal and empirical analysis of Helmert et al. (2007), this flexibility often makes the merge-and-shrink abstractions much more effective than PDBs. However, the merge-and-shrink abstract spaces are still searched explicitly, and thus they still have to be of fixed size. While quality heuristics estimates can still be obtained for many problems, this limitation is a critical obstacle for many others.

Our goal in this chapter is to push the envelope of abstraction heuristics beyond explicit abstractions. We introduce a principled way to obtain abstraction heuristics that limit neither the dimensionality nor the size of the abstract spaces. The basic idea behind what we call *implicit abstractions* is simple and intuitive: instead of relying on abstract problems that are easy to solve because they are small, we can rely on abstract problems belonging to provably tractable fragments of optimal planning. The key point is that, at least theoretically, moving to implicit abstractions removes the requirement on the abstractions size to be small. Our contribution, however, is in showing that implicit abstractions are far from being of theoretical interest only. Specifically,

1. We specify *acyclic causal-graph decompositions*, a general framework for additive implicit abstractions that is based on decomposing the problem at hand along its causal graph. We then introduce a concrete family of such abstractions, called *fork decompositions*, that are based on two novel fragments of tractable cost-optimal planning. Following the type of analysis suggested by Helmert and Mattmüller (2008), we formally analyze the asymptotic performance ratio of the fork-decomposition heuristics and prove that their worst-case accuracy on selected domains is comparable with that of (even parametric) state-of-the-art admissible heuristics. We then empirically evaluate the accuracy of the fork-decomposition heuristics on a large set of domains from recent planning competitions and show that their accuracy is competitive with the state of the art.

[1] This does not necessarily apply to *symbolic* PDBs which, on some tasks, may exponentially reduce the PDB's representation (Edelkamp, 2002).

2. The key attraction of explicit abstractions is that costs in the abstract space can be precomputed and stored in memory in a preprocessing phase so that heuristic evaluation during search can be done by a simple lookup. A necessary condition for this would seem to be the small size of the abstract space. However, we show that an equivalent of the PDB and merge-and-shrink's notion of "database" exists for the fork-decomposition abstractions as well, despite the exponential-size abstract spaces of the latter. These *databased implicit abstractions* are based on a proper partitioning of the heuristic computation into parts that can be shared between search states and parts that must be computed online per state. Our empirical evaluation shows that A^* equipped with the "databased" fork-decomposition heuristics favorably competes with the state of the art of cost-optimal planning.

This work is a revision and extension of the formulation and results presented by Katz and Domshlak (2008c, 2009b), which in turn is based on ideas first sketched also by Katz and Domshlak (2007a).

4.1 Implicit Abstractions: Basic Idea

Focusing on the $O(1)$ bound posted by explicit abstractions on the size of the abstract space, our first observation is that explicit abstractions are not necessarily the only way to proceed with abstraction heuristics. Given a planning task Π over states S, suppose we can transform it into a different planning task Π^α such that

1. the transformation induces an abstraction mapping $\alpha : S \to S^\alpha$ where S^α is the state space of Π^α, and

2. both the transformation of Π to Π^α, as well as computing α for any state in S, can be done in time polynomial in $||\Pi||$.

Having such planning-task-to-planning-task transformations in mind, we define what we call (additive) *implicit abstractions*.

Definition 9 *An* **additive implicit abstraction** *of a planning task Π is a set of pairs $\mathcal{A} = \{\langle \Pi_i, \alpha_i \rangle\}_{i=1}^k$ such that $\{\Pi_i\}_{i=1}^k$ are some planning tasks and $\{\langle \mathcal{T}(\Pi_i), \alpha_i \rangle\}_{i=1}^k$ is an additive abstraction of $\mathcal{T}(\Pi)$.*

Let us now examine the notion of implicit abstractions more closely. First, implicit abstractions allow for a natural additive combination of admissible heuristics for the abstract tasks. This composition is formulated below by Theorem 6, extending the original criterion for admissibility of additive heuristics described in Section 2.1. Second, as formulated by Theorem 7, implicit abstractions can be composed via the functional composition of their abstraction mappings. These two easy-to-prove properties of implicit abstractions allow us then to take the desired step from implicit abstractions to implicit abstraction heuristics.

Theorem 6 (Admissibility) *Let Π be a planning task and $\mathcal{A} = \{\langle \Pi_i, \alpha_i \rangle\}_{i=1}^k$ be an additive implicit abstraction of Π. If, for each $1 \leq i \leq k$, h_i is an admissible heuristic for Π_i, then the function $h(s) = \sum_{i=1}^k h_i(\alpha_i(s))$ is an admissible heuristic for Π.*

Proof: The proof is straightforward. Let $\mathcal{T} = (S, L, Tr, s^I, S^G, \varpi)$ be the transition graph of Π, and let s be some state in S. For each $1 \leq i \leq k$, let $\mathcal{T}_i = (S_i, L_i, Tr_i, s_i^I, S_i^G, \varpi_i)$ be the transition graph of Π_i.

First, if h_i is an admissible heuristic for Π_i, then for all $s_i \in S_i^G$,

$$h_i(\alpha_i(s)) \leq cost(\alpha_i(s), s_i).$$

Now, for each state $s' \in S^G$, from Definition 3 we have $\alpha_i(s') \in S_i^G$, and from Eq. 2.1 we have

$$\sum_{i=1}^{k} cost(\alpha_i(s), \alpha_i(s')) \leq cost(s, s'),$$

and thus

$$h(s) = \sum_{i=1}^{k} h_i(\alpha_i(s)) \leq \sum_{i=1}^{k} cost(\alpha_i(s), \alpha_i(s')) \leq cost(s, s'),$$

giving us an admissible estimate for $h^*(s)$. ∎

Theorem 7 (Composition) *Let Π be a planning task and $\mathcal{A} = \{\langle \Pi_i, \alpha_i \rangle\}_{i=1}^{k}$ be an additive implicit abstraction of Π. If, for each $1 \leq i \leq k$, $\mathcal{A}_i = \{\langle \Pi_{i,j}, \alpha_{i,j} \rangle\}_{j=1}^{k_i}$ is an additive implicit abstraction of Π_i, then $\mathcal{A}' = \bigcup_{i=1}^{k} \{\langle \Pi_{i,j}, \alpha_{i,j} \circ \alpha_i \rangle\}_{j=1}^{k_i}$ is an additive implicit abstraction of Π.*

Proof: Let $\mathcal{T} = (S, L, Tr, s^I, S^G, \varpi)$ be the transition graph of Π. For each $1 \leq i \leq k$, let $\mathcal{T}_i = (S_i, L_i, Tr_i, s_i^I, S_i^G, \varpi_i)$ be the transition graph of Π_i, and for each $1 \leq j \leq k_i$, let $\mathcal{T}_{i,j} = (S_{i,j}, L_{i,j}, Tr_{i,j}, s_{i,j}^I, S_{i,j}^G, \varpi_{i,j})$ be the transition graph of $\Pi_{i,j}$. We need to show that $\alpha_{i,j} \circ \alpha_i$ is an abstraction mapping as in Definition 3. From α_i and $\alpha_{i,j}$ being abstraction mappings, we have

- $s_{i,j}^I = \alpha_{i,j}(s_i^I) = \alpha_{i,j}(\alpha_i(s^I)) = \alpha_{i,j} \circ \alpha_i(s^I)$,
- for all $s \in S^G$ we have $\alpha_i(s) \in S_i^G$ and thus $\alpha_{i,j}(\alpha_i(s)) = \alpha_{i,j} \circ \alpha_i(s) \in S_{i,j}^G$, and
- for all $s_i, s_i' \in S_i$, $cost(s_i, s_i') \geq \sum_{j=1}^{k_i} cost(\alpha_{i,j}(s_i), \alpha_{i,j}(s_i'))$, and thus for all $s, s' \in S$,

$$\begin{aligned} cost(s, s') \geq \sum_{i=1}^{k} cost(\alpha_i(s), \alpha_i(s')) &\geq \sum_{i=1}^{k} \sum_{j=1}^{k_i} cost(\alpha_{i,j}(\alpha_i(s)), \alpha_{i,j}(\alpha_i(s'))) \\ &= \sum_{i=1}^{k} \sum_{j=1}^{k_i} cost(\alpha_{i,j} \circ \alpha_i(s), \alpha_{i,j} \circ \alpha_i(s')). \end{aligned}$$

∎

Together, Theorems 6 and 7 suggest the following scheme for deriving abstraction heuristics. Given an additive implicit abstraction $\mathcal{A} = \{\langle \Pi_i, \alpha_i \rangle\}_{i=1}^{k}$, if all its individual abstract tasks belong to some *tractable fragments of optimal planning*, then we can use in practice the (sum of the) true costs in all Π_i as the admissible estimates for the costs in Π. Otherwise, if optimal planning for some abstract tasks Π_i in $\mathcal{A}$ cannot be proven polynomial-time solvable, then we can further abstract these tasks, obtaining admissible estimates for the true costs in Π_i.

Definition 10 *Let Π be a planning task over states S, and let $\mathcal{A} = \{\langle \Pi_i, \alpha_i \rangle\}_{i=1}^k$ be an additive implicit abstraction of Π. If $k = O(poly(||\Pi||))$, and, for all states $s \in S$ and all $1 \leq i \leq k$, $h^*(\alpha_i(s))$ is polynomial-time computable, then $h^{\mathcal{A}}(s) = \sum_{i=1}^k h^*(\alpha_i(s))$ is an* **implicit abstraction heuristic** *function for Π.*

Compared to explicit abstraction heuristics such as PDB heuristics and merge-and-shrink heuristics, the direction of implicit abstraction heuristics is, at least in principle, appealing because neither the dimensionality nor even the size of the state spaces induced by implicit abstractions are required to be bounded by something restrictive, if at all. The pitfall, however, is that implicit abstraction heuristics correspond to tractable fragments of optimal planning, and the palette of such known fragments is extremely limited (Bäckström & Nebel, 1995; Bylander, 1994; Jonsson & Bäckström, 1998a; Jonsson, 2007; Katz & Domshlak, 2007b). In fact, none so far has appeared to us very convenient for automatically devising useful problem transformations as above. Fortunately, we show next that the boundaries of tractability can be expanded in the right way, allowing us to successfully materialize the idea of implicit abstraction heuristics.

4.2 Acyclic Causal-Graph Decompositions

In the following, a key role is played by the causal graphs induced by the planning tasks. Informally, the basic idea behind what we call *causal-graph decompositions* is to abstract the given planning task Π along a subgraph of Π's causal graph, with the goal of obtaining abstract problems of *specific structure*. Naturally, there are numerous possibilities for obtaining such structure-oriented abstractions. We now present one such decomposition that is tailored to abstractions around *acyclic subgraphs*. Informally, this decomposition can be seen as a sequential application of two kinds of task transformations: dropping preconditions (Pearl, 1984) and (certain form of) breaking actions with conjunctive effects into single-effect actions.

Definition 11 *Let $\Pi = \langle V, A, I, G, cost \rangle$ be a planning task, and let $\mathcal{G} = (V_\mathcal{G}, E_\mathcal{G})$ be an acyclic subgraph of the causal graph $CG(\Pi)$. A planning task $\Pi_\mathcal{G} = \langle V_\mathcal{G}, A_\mathcal{G}, I_\mathcal{G}, G_\mathcal{G}, cost_\mathcal{G} \rangle$ is an* **acyclic causal-graph decomposition** *of Π with respect to $\mathcal{G}$ if*

1. *$I_\mathcal{G} = I[V_\mathcal{G}]$, $G_\mathcal{G} = G[V_\mathcal{G}]$,*
2. *$A_\mathcal{G} = \bigcup_{a \in A} A_\mathcal{G}(a)$ where each $A_\mathcal{G}(a) = \{a^1, \ldots, a^{l(a)}\}$ is a set of actions over $V_\mathcal{G}$ such that, for a topological with respect to $\mathcal{G}$ ordering of the variables $\{v_1, \ldots, v_{l(a)}\} = \mathcal{V}(\mathsf{eff}(a)) \cap V_\mathcal{G}$, and $1 \leq i \leq l(a)$,*

$$\mathsf{eff}(a^i)[v] = \begin{cases} \mathsf{eff}(a)[v], & v = v_i \\ \text{unspecified}, & \text{otherwise} \end{cases}$$

$$\mathsf{pre}(a^i)[v] = \begin{cases} \mathsf{pre}(a)[v], & (v, v_i) \in E_\mathcal{G} \wedge v \notin \mathcal{V}(\mathsf{eff}(a)) \text{ or } v = v_i \\ \mathsf{eff}(a)[v], & (v, v_i) \in E_\mathcal{G} \wedge v \in \mathcal{V}(\mathsf{eff}(a)) \\ \text{unspecified}, & \text{otherwise} \end{cases} \tag{4.1}$$

3. *For each action $a \in A$,*

$$\sum_{a' \in A_\mathcal{G}(a)} cost_\mathcal{G}(a') \leq cost(a). \tag{4.2}$$

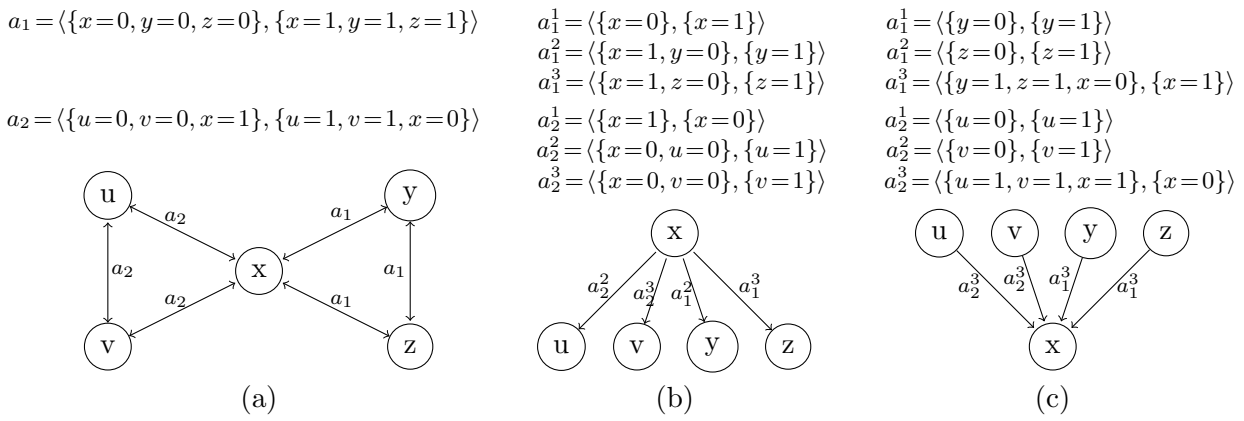


Figure 4.1: (a) The actions and causal graph $CG(\Pi)$ of the planning graph in the example illustrating Definition 11. (b) Subgraph $\mathcal{G}_1$ of $CG(\Pi)$ and the induced action sets $A_{\mathcal{G}_1}(a_1)$ and $A_{\mathcal{G}_1}(a_2)$. (c) Subgraph $\mathcal{G}_2$ of $CG(\Pi)$ and the induced action sets $A_{\mathcal{G}_2}(a_1)$ and $A_{\mathcal{G}_2}(a_2)$. The arcs of both $CG(\Pi)$ and its subgraphs $\mathcal{G}_1$ and $\mathcal{G}_2$ are labeled with the actions inducing the arcs.

It is not hard to verify from Definition 11 that for any planning task Π and any acyclic causal-graph decomposition $\Pi_{\mathcal{G}}$ of Π, the causal graph $CG(\Pi_{\mathcal{G}})$ is exactly the subgraph $\mathcal{G}$ underlying the decomposition. To illustrate the notion of acyclic causal-graph decomposition, we consider a planning task $\Pi = \langle V, A, I, G, cost\rangle$ over five state variables $V = \{u, v, x, y, z\}$, two unit-cost actions $A = \{a_1, a_2\}$ as in Figure 4.1a, initial state $I = \{u = 0, v = 0, x = 0, y = 0, z = 0\}$, and goal $G = \{u = 1, v = 1, x = 0, y = 1, z = 1\}$. The causal graph $CG(\Pi)$ is depicted in Figure 4.1a. Figures 4.1b-c show two subgraphs $\mathcal{G}_1$ and $\mathcal{G}_2$ of $CG(\Pi)$, respectively, as well as the action sets $A_{\mathcal{G}_1}(a_1) = \{a_1^1, a_1^2, a_1^3\}$ and $A_{\mathcal{G}_1}(a_2) = \{a_2^1, a_2^2, a_2^3\}$ in Figure 4.1(b), and the action sets $A_{\mathcal{G}_2}(a_1) = \{a_1^1, a_1^2, a_1^3\}$ and $A_{\mathcal{G}_2}(a_2) = \{a_2^1, a_2^2, a_2^3\}$ in Figure 4.1(c). For $i \in \{1, 2\}$, let $\Pi_i = \langle V, A_i, I, G, cost_i\rangle$ be the planning task with $A_i = A_{\mathcal{G}_i}(a_1) \cup A_{\mathcal{G}_i}(a_2)$ and $cost_i(a) = 1/3$ for all $a \in A_i$. These two planning tasks Π_i (individually) satisfy the conditions of Definition 11 with respect to Π and $\mathcal{G}_i$, and thus they are acyclic causal-graph decompositions of Π with respect to $\mathcal{G}_i$.

We now proceed with specifying implicit abstractions defined via acyclic causal-graph decompositions.

Definition 12 *Let $\Pi = \langle V, A, I, G, cost\rangle$ be a planning task over states S, and let $\mathbf{G} = \{\mathcal{G}_i = (V_{\mathcal{G}_i}, E_{\mathcal{G}_i})\}_{i=1}^k$ be a set of acyclic subgraphs of the causal graph $CG(\Pi)$. $\mathcal{A} = \{\langle \Pi_{\mathcal{G}_i}, \alpha_i\rangle\}_{i=1}^k$ is an* **acyclic causal-graph abstraction** *of Π over $\mathbf{G}$ if, for some set of cost functions $\{cost_i : A \to \mathbb{R}^{0+}\}_{i=1}^k$ satisfying*

$$\forall a \in A: \quad \sum_{i=1}^{m} cost_i(a) \leq cost(a), \tag{4.3}$$

we have, for $1 \leq i \leq k$,

- *$\Pi^{\mathcal{G}_i} = \langle V^{\mathcal{G}_i}, A^{\mathcal{G}_i}, I^{\mathcal{G}_i}, G^{\mathcal{G}_i}, cost^{\mathcal{G}_i}\rangle$ is an acyclic causal-graph decomposition of $\Pi_i = \langle V, A, I, G, cost_i\rangle$ with respect to $\mathcal{G}_i$, and*
- *the abstraction mapping $\alpha_i : S \to S_i$ is the projection mapping $\alpha_i(s) = s[V_{\mathcal{G}_i}]$.*

Theorem 8 *Acyclic causal-graph abstractions of the planning tasks are additive implicit abstractions of these tasks.*

Proof: Let $\Pi = \langle V, A, I, G, cost\rangle$ be a planning task, and let $\mathcal{A} = \{\langle \Pi_{\mathcal{G}_i}, \alpha_i\rangle\}_{i=1}^{k}$ be an acyclic causal-graph abstraction of Π over a sets of subgraphs $\mathbf{G} = \{\mathcal{G}_i = (V_{\mathcal{G}_i}, E_{\mathcal{G}_i})\}_{i=1}^{k}$. Let $\mathcal{T} = (S, L, Tr, s^I, S^G, \varpi)$ be the transition graph of Π, and, for $1 \leq i \leq k$, $\mathcal{T}_i = (S_i, L_i, Tr_i, s_i^I, S_i^G, \varpi_i)$ be the transition graph of $\Pi_{\mathcal{G}_i}$. We need to show that α_i is an abstraction mapping as in Definition 3.

First, from Definitions 11 and 12, we have

- $s_i^I = I_{\mathcal{G}_i} = I[V_{\mathcal{G}_i}] = s^I[V_{\mathcal{G}_i}] = \alpha_i(s^I)$, and
- for all $s \in S^G$ we have $s \supseteq G$ and thus $\alpha_i(s) = s[V_{\mathcal{G}_i}] \supseteq G[V_{\mathcal{G}_i}] = G_{\mathcal{G}_i}$, providing us with $\alpha_i(s) \in S_i^G$.

Now, if s is a state of Π and $a \in A$ is an action with $\mathsf{pre}(a) \subseteq s$, then $\alpha_i(s)$ is a state of $\Pi_{\mathcal{G}_i}$ and $\mathsf{pre}(a)[V_{\mathcal{G}_i}] \subseteq \alpha_i(s)$. Let the action sequence $\rho = \langle a^1, a^2, \ldots, a^{l(a)}\rangle$ be constructed from a as in Eq. 4.1. We inductively prove that ρ is applicable in $\alpha_i(s)$. First, for each $v \in V_{\mathcal{G}_i}$, either $\mathsf{pre}(a^1)[v] = \mathsf{pre}(a)[v]$, or $\mathsf{pre}(a^1)[v]$ is unspecified, and thus $\rho_1 = \langle a^1\rangle$ is applicable in $\alpha_i(s)$. The inductive hypothesis is now that $\rho_j = \langle a^1, a^2, \ldots, a^j\rangle$ is applicable in $\alpha_i(s)$, and let $s' = \alpha_i(s)[\![\rho_j]\!]$. From Eq. 4.1, for each $1 \leq j' \leq j$, $a^{j'}$ changes the value of $v_{j'}$ to $\mathsf{eff}(a)[v_{j'}]$, and that is the only change of $v_{j'}$ along ρ_j. Likewise, since all the actions constructed as in Eq. 4.1 are unary-effect, $\{v_1, \ldots, v_j\}$ are the only variables in $V_{\mathcal{G}_i}$ affected along ρ_j. Hence, for all $v \in V_{\mathcal{G}_i}$, if $v = v_{j'}$, $1 \leq j' \leq j$, then $s'[v] = \mathsf{eff}(a)[v] = \mathsf{pre}(a^{j+1})[v]$, and otherwise, $s'[v] = \alpha_i(s)[v]$, and if $\mathsf{pre}(a^{j+1})[v]$ is specified, then $\mathsf{pre}(a^{j+1})[v] = \mathsf{pre}(a)[v] = \alpha_i(s)[v]$. This implies that a^{j+1} is applicable in s' and, as a result, $\rho_{j+1} = \langle a^1, a^2, \ldots, a^{j+1}\rangle$ is applicable in $\alpha_i(s)$, finalizing the inductive proof. Likewise, exactly the same arguments on the affect of $\{a^j\}_{j=1}^{l(a)}$ on $\alpha_i(s)$ immediately imply that, if $\rho = \langle a^1, a^2, \ldots, a^{l(a)}\rangle$, then $\alpha_i(s[\![a]\!]) = \alpha_i(s)[\![\rho]\!]$.

Next, for each $a \in A$, from Eqs. 4.2 and 4.3 we have

$$\sum_{i=1}^{k} \sum_{a' \in A_{\mathcal{G}_i}(a)} cost_{\mathcal{G}_i}(a') \leq \sum_{i=1}^{k} cost_i(a) \leq cost(a). \tag{4.4}$$

Now, let $s, s' \in S$ be a pair of original states such that $cost(s, s') < \infty$, and let $\varrho = \langle a_1, \ldots, a_k\rangle$ be the sequence of labels along a shortest path from s to s' in $\mathcal{T}$. From that, $cost(s, s') = cost(\varrho) = \sum_{j=1}^{k} cost(a_j)$. The decomposition of such a path to the sequences of actions as in Eq. 4.1 is a (not neccesarily shortest) path from $\alpha_i(s)$ to $\alpha_i(s')$ in $\mathcal{T}_i$, and thus $cost(\alpha_i(s), \alpha_i(s')) \leq \sum_{j=1}^{k} \sum_{a' \in A_{\mathcal{G}_i}(a_j)} cost_{\mathcal{G}_i}(a')$, providing us with

$$\begin{aligned}\sum_{i=1}^{k} cost(\alpha_i(s), \alpha_i(s')) &\leq \sum_{i=1}^{k}\sum_{j=1}^{k} \sum_{a' \in A_{\mathcal{G}_i}(a_j)} cost_{\mathcal{G}_i}(a') = \sum_{j=1}^{k}\sum_{i=1}^{k} \sum_{a' \in A_{\mathcal{G}_i}(a_j)} cost_{\mathcal{G}_i}(a') \overset{(4.4)}{\leq} \sum_{j=1}^{k} cost(a_j) \\ &= cost(s, s').\end{aligned}$$

■

Thus, if we can decompose the given task Π into a set of tractable acyclic causal-graph decompositions $\mathbf{\Pi} = \{\Pi_{\mathcal{G}_1}, \ldots, \Pi_{\mathcal{G}_m}\}$, then we can solve all these tasks in polynomial time, and derive

an additive admissible heuristic for Π. Before we proceed with considering concrete acyclic causal-graph decomposition, note that Definition 3 leaves the decision about the actual partition of the action costs rather open. In what follows we adopt the most straightforward, *uniform* action cost partition in which the cost of each action a is *equally split* among all the non-redundant representatives of a in $\bigcup_{i=1}^{m} A_{\mathcal{G}_i}(a)$. However, a better choice of action cost partition can sometimes be made. In fact, sometimes it can even be optimized; for further details see Chapter 5.

4.3 Fork Decomposition

We now proceed with introducing two concrete acyclic causal-graph decompositions that, when combined with certain variable domain abstractions, provide us with implicit abstraction heuristics. These so called *fork-decomposition heuristics* are based on two novel fragments of tractable cost-optimal planning for tasks with fork and inverted-fork structured causal graphs.

Definition 13 *For a planning task Π over variables V, and a variable $v \in V$,*

(1) **v-fork** *of Π is the subgraph $\mathcal{G}^{\mathsf{f}}_v$ of $CG(\Pi)$ over nodes $V_{\mathcal{G}^{\mathsf{f}}_v} = \{v\} \cup \mathsf{succ}(v)$ and edges $E_{\mathcal{G}^{\mathsf{f}}_v} = \{(v,u) \mid u \in \mathsf{succ}(v)\}$, and*

(2) **v-ifork** *(short for inverted fork) of Π is a subgraph $\mathcal{G}^{\mathsf{i}}_v$ of $CG(\Pi)$ over nodes $V_{\mathcal{G}^{\mathsf{i}}_v} = \{v\} \cup \mathsf{pred}(v)$ and edges $E_{\mathcal{G}^{\mathsf{i}}_v} = \{(u,v) \mid u \in \mathsf{pred}(v)\}$.*

The sets of all v-forks and all v-iforks of Π are denoted by $\mathbf{G}_{\mathcal{F}} = \{\mathcal{G}^{\mathsf{f}}_v\}_{v \in V}$ and $\mathbf{G}_{\mathcal{I}} = \{\mathcal{G}^{\mathsf{i}}_v\}_{v \in V}$, respectively.

For any planning task and each of its state variables v, both v-fork and v-ifork are acyclic digraphs, allowing us to define our three implicit abstractions as follows.

Definition 14 *For any planning task $\Pi = \langle V, A, I, G, cost \rangle$,*

(1) any acyclic causal-graph abstraction $\mathcal{A}_{\mathcal{F}} = \{\langle \Pi^{\mathsf{f}}_v, \alpha^{\mathsf{f}}_v \rangle\}_{v \in V}$ of Π over $\mathbf{G}_{\mathcal{F}}$ is called **$\mathcal{F}$-abstraction**, *and the set of abstract planning tasks $\mathbf{\Pi}_{\mathcal{F}} = \{\Pi^{\mathsf{f}}_v\}_{v \in V}$ is called* **$\mathcal{F}$-decomposition** *of Π;*

(2) any acyclic causal-graph abstraction $\mathcal{A}_{\mathcal{I}} = \{\langle \Pi^{\mathsf{i}}_v, \alpha^{\mathsf{i}}_v \rangle\}_{v \in V}$ of Π over $\mathbf{G}_{\mathcal{I}}$ is called **$\mathcal{I}$-abstraction**, *and the set of abstract planning tasks $\mathbf{\Pi}_{\mathcal{I}} = \{\Pi^{\mathsf{i}}_v\}_{v \in V}$ is called* **$\mathcal{I}$-decomposition** *of Π;*

(3) any acyclic causal-graph abstraction $\mathcal{A}_{\mathcal{FI}} = \{\langle \Pi^{\mathsf{f}}_v, \alpha^{\mathsf{f}}_v \rangle, \langle \Pi^{\mathsf{i}}_v, \alpha^{\mathsf{i}}_v \rangle\}_{v \in V}$ of Π over $\mathbf{G}_{\mathcal{FI}} = \mathbf{G}_{\mathcal{F}} \cup \mathbf{G}_{\mathcal{I}}$ is called **$\mathcal{FI}$-abstraction**, *and the set of abstract planning tasks $\mathbf{\Pi}_{\mathcal{FI}} = \{\Pi^{\mathsf{f}}_v, \Pi^{\mathsf{i}}_v\}_{v \in V}$ is called* **$\mathcal{FI}$-decomposition** *of Π.*

Definition 14 can be better understood by considering the $\mathcal{FI}$-abstraction of the problem Π from our Logistics example; Figure 4.2 schematically illustrates the process. To simplify the example, here we as if eliminate from $\mathbf{G}_{\mathcal{FI}}$ all the single-node subgraphs, obtaining

$$\mathcal{A}_{\mathcal{FI}} = \{\langle \Pi^{\mathsf{f}}_{c_1}, \alpha^{\mathsf{f}}_{c_1} \rangle, \{\langle \Pi^{\mathsf{f}}_{c_2}, \alpha^{\mathsf{f}}_{c_2} \rangle, \{\langle \Pi^{\mathsf{f}}_{c_3}, \alpha^{\mathsf{f}}_{c_3} \rangle, \{\langle \Pi^{\mathsf{f}}_{t}, \alpha^{\mathsf{f}}_{t} \rangle, \{\langle \Pi^{\mathsf{i}}_{p_1}, \alpha^{\mathsf{i}}_{p_1} \rangle, \{\langle \Pi^{\mathsf{i}}_{p_2}, \alpha^{\mathsf{i}}_{p_2} \rangle\}.$$

Considering the action sets of the problems in $\mathbf{\Pi}_{\mathcal{FI}} = \{\Pi^{\mathsf{f}}_{c_1}, \Pi^{\mathsf{f}}_{c_2}, \Pi^{\mathsf{f}}_{c_3}, \Pi^{\mathsf{f}}_{t}, \Pi^{\mathsf{i}}_{p_1}, \Pi^{\mathsf{i}}_{p_2}\}$, we see that each original driving action has one nonredundant (that is, "changing some variable") representative in

three of the abstract planning tasks, while each load/unload action has one nonredundant representative in five of these tasks. For instance, the action *drive-c_1-from-A-to-D* has one nonredundant representative in each of the tasks $\{\Pi^{\mathsf{f}}_{c_1}, \Pi^{\mathsf{i}}_{p_1}, \Pi^{\mathsf{i}}_{p_2}\}$, and the action *load-$p_1$-into-$c_1$-at-A* has one nonredundant representative in each of the tasks $\{\Pi^{\mathsf{f}}_{c_1}, \Pi^{\mathsf{f}}_{c_2}, \Pi^{\mathsf{f}}_{c_3}, \Pi^{\mathsf{f}}_{t}, \Pi^{\mathsf{i}}_{p_1}\}$. Since we assume a uniform partition of the action costs, the cost of each driving and load/unload action in each relevant abstract planning task is thus set to 1/3 and 1/5, respectively. From Theorem 8 we have $\mathcal{A}_{\mathfrak{FJ}}$ being an additive implicit abstraction of Π, and from Theorem 6 we then have

$$h^{\mathfrak{FJ}} = \sum_{v \in V} \Big(h^*_{\Pi^{\mathsf{f}}_v} + h^*_{\Pi^{\mathsf{i}}_v} \Big), \tag{4.5}$$

being an admissible estimate of h^* in Π. The question now is how good this estimate is. The optimal cost of solving our running example is 19. Taking as a reference the well-known admissible heuristics $h_{\max}$ (Bonet & Geffner, 2001) and h^2 (Haslum & Geffner, 2000), we have $h_{\max}(I) = 8$ and $h^2(I) = 13$. Considering our $\mathcal{FJ}$-abstraction, the optimal plans for the tasks in $\mathbf{\Pi}_{\mathfrak{FJ}}$ are as follows.

$\Pi^{\mathsf{f}}_{c_1}$: *load-p_1-into-c_2-at-C, unload-p_1-from-c_2-at-D, load-p_1-into-t-at-D, unload-p_1-from-t-at-E, load-p_1-into-c_3-at-E, unload-p_1-from-c_3-at-G, load-p_2-into-c_3-at-F, unload-p_2-from-c_3-at-E.*

$\Pi^{\mathsf{f}}_{c_2}$: *load-p_1-into-c_1-at-C, unload-p_1-from-c_1-at-D, load-p_1-into-t-at-D, unload-p_1-from-t-at-E, load-p_1-into-c_3-at-E, unload-p_1-from-c_3-at-G, load-p_2-into-c_3-at-F, unload-p_2-from-c_3-at-E.*

$\Pi^{\mathsf{f}}_{c_3}$: *load-p_1-into-c_1-at-C, unload-p_1-from-c_1-at-D, load-p_1-into-t-at-D, unload-p_1-from-t-at-E, drive-c_3-from-G-to-E, load-p_1-into-c_3-at-E, drive-c_3-from-E-to-G, unload-p_1-from-c_3-at-G, drive-c_3-from-G-to-E, drive-c_3-from-E-to-F, load-p_2-into-c_3-at-F, drive-c_3-from-F-to-E, unload-p_2-from-c_3-at-E, drive-c_3-from-E-to-F.*

Π^{f}_{t} : *load-p_1-into-c_1-at-C, unload-p_1-from-c_1-at-D, drive-t-from-E-to-D, load-p_1-into-t-at-D, drive-t-from-D-to-E, unload-p_1-from-t-at-E, load-p_1-into-c_3-at-E, unload-p_1-from-c_3-at-G, load-p_2-into-c_3-at-F, unload-p_2-from-c_3-at-E.*

$\Pi^{\mathsf{i}}_{p_1}$: *drive-c_1-from-A-to-D, drive-c_1-from-D-to-C, load-p_1-into-c_1-at-C, drive-c_1-from-C-to-D, unload-p_1-from-c_1-at-D, drive-t-from-E-to-D, load-p_1-into-t-at-D, drive-t-from-D-to-E, unload-p_1-from-t-at-E, drive-c_3-from-G-to-E, load-p_1-into-c_3-at-E, drive-c_3-from-E-to-G, unload-p_1-from-c_3-at-G, drive-c_3-from-G-to-E, drive-c_3-from-E-to-F.*

$\Pi^{\mathsf{i}}_{p_2}$: *drive-c_3-from-G-to-E, drive-c_3-from-E-to-F, load-p_2-into-c_3-at-F, drive-c_3-from-F-to-E, unload-p_2-from-c_3-at-E, drive-c_3-from-E-to-F.*

Hence, we have

$$\begin{array}{rcccccccccccl} h^{\mathfrak{FJ}} & = & h^*_{\Pi^{\mathsf{f}}_{c_1}} & + & h^*_{\Pi^{\mathsf{f}}_{c_2}} & + & h^*_{\Pi^{\mathsf{f}}_{c_3}} & + & h^*_{\Pi^{\mathsf{f}}_{t}} & + & h^*_{\Pi^{\mathsf{i}}_{p_1}} & + & h^*_{\Pi^{\mathsf{f}}_{p_2}} \\ & = & \frac{8}{5} & + & \frac{8}{5} & + & \frac{8}{5}+\frac{6}{3} & + & \frac{8}{5}+\frac{2}{3} & + & \frac{6}{5}+\frac{9}{3} & + & \frac{2}{5}+\frac{4}{3} & = & 15, \end{array} \tag{4.6}$$

and so $h^{\mathfrak{FJ}}$ appears at least promising.

Unfortunately, despite the seeming simplicity of the planning tasks in $\mathbf{\Pi}_{\mathfrak{FJ}}$, it turns out that implicit fork-decomposition abstractions as in Definitions 14 do not fit the requirements of implicit abstraction heuristics as in Definition 10. The causal graphs of the planning tasks in $\mathbf{\Pi}_{\mathfrak{F}}$ and

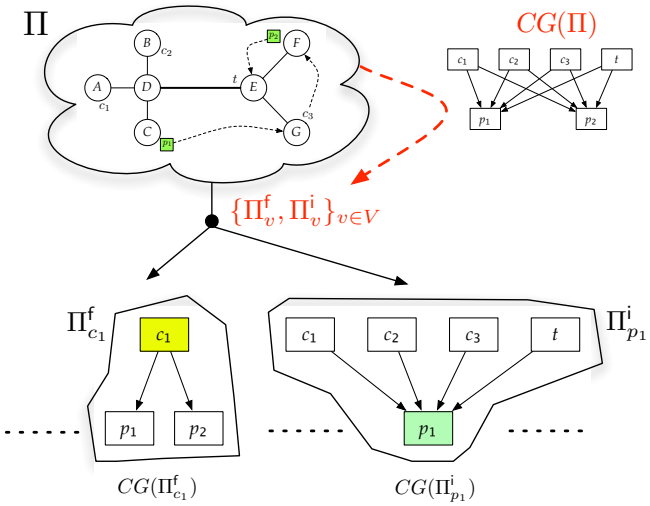


Figure 4.2: Schematic illustration of $\mathcal{FJ}$-decomposition for our running Logistics example

$\mathbf{\Pi}_{\mathcal{J}}$ form directed forks and directed inverted forks, respectively, and, in general, the number of variables in each such planning task can be as large as $\Theta(|V|)$. The problem is that even satisficing planning for SAS$^+$ fragments with fork and inverted fork causal graphs is NP-complete (Domshlak & Dinitz, 2001). In fact, recent results by Chen and Giménez (2008) show that planning for any SAS$^+$ fragment characterized by any nontrivial form of causal graph is NP-hard. Moreover, even if the domain transition graphs of all the state variables are strongly connected (as in our example), optimal planning for fork and inverted fork structured problems remain NP-hard (see Helmert 2003 and 2004 for the respective results). Next, however, we show that this is not the end of the story for fork decompositions.

While the hardness of optimal planning for problems with fork and inverted fork causal graphs casts a shadow on the relevance of fork decompositions, a closer look at the proofs of the corresponding hardness results of Domshlak and Dinitz (2001) and Helmert (2003, 2004) reveals that they in particular rely on root variables having large domains. Exploiting this observation, we show that this reliance is not incidental and characterize two substantial islands of tractability within the structural fragments of SAS$^+$. Theorems 4 and 5 in Chapter 3 clarify the gap between fork decompositions and implicit abstraction heuristics, and now we can bridge this gap by further abstracting each task in the given fork decomposition of Π. We do that by abstracting domains of the fork roots and inverted-fork sinks to meet the requirements of the tractable fragments. We note that, in itself, the idea of domain decomposition is not very new in general (Hernadvölgyi & Holte, 1999) and in domain-independent planning in particular (Domshlak, Hoffmann, & Sabharwal, 2009). In fact, the shrinking step of the algorithm for building the merge-and-shrink abstractions is precisely a variable domain abstraction for meta-variables constructed in the merging steps (Helmert et al., 2007).

Definition 15 *Let $\Pi = \langle V, A, I, G, cost\rangle$ be a planning task over states S, $v \in V$ be a state variable, and $\Phi = \{\phi_1, \ldots, \phi_k\}$ be a set of mappings from $\mathcal{D}(v)$ to some sets $\Gamma_1, \ldots, \Gamma_k$, respectively.*

$\mathcal{A} = \{\langle \Pi_{\phi_i}, \alpha_i \rangle\}_{i=1}^k$ *is a* **domain abstraction** *of* Π *over* Φ *if, for some set of cost functions* $\{cost_i : A \to \mathbb{R}^{0+}\}_{i=1}^k$ *satisfying*

$$\forall a \in A: \quad \sum_{i=1}^{m} cost_i(a) \leq cost(a), \tag{4.7}$$

we have, for $1 \leq i \leq k$,

- *the abstraction mapping* α_i *of states* S *is*

$$\forall u \in V: \quad \alpha_i(s)[u] = \begin{cases} \phi_i(s[u]), & u = v \\ s[u], & u \neq v \end{cases},$$

  and, extending α_i *to partial assignments on* $V' \subseteq V$ *as* $\alpha_i(s[V']) = \alpha_i(s)[V']$,

- $\Pi_{\phi_i} = \langle V, A_{\phi_i}, I_{\phi_i}, G_{\phi_i}, cost_{\phi_i} \rangle$ *is a planning task with*

  1. $I_{\phi_i} = \alpha_i(I)$, $G_{\phi_i} = \alpha_i(G)$,
  2. $A_{\phi_i} = \{a_{\phi_i} = \langle \alpha_i(\mathsf{pre}(a)), \alpha_i(\mathsf{eff}(a)) \rangle \mid a \in A\}$, *and*
  3. *for each action* $a \in A$,

$$cost_{\phi_i}(a_{\phi_i}) = cost_i(a). \tag{4.8}$$

We say that Π_{ϕ_i} *is a* **domain decomposition** *of* $\Pi_i = \langle V, A, I, G, cost_i \rangle$ *with respect to* ϕ_i.

Theorem 9 *Domain abstractions of the planning tasks are additive implicit abstractions of these tasks.*

Proof: Let $\Pi = \langle V, A, I, G, cost \rangle$ be a planning task and $\mathcal{A} = \{\langle \Pi_{\phi_i}, \alpha_i \rangle\}_{i=1}^k$ be a domain abstraction of Π over $\Phi = \{\phi_1, \ldots, \phi_k\}$. Let $\mathcal{T} = (S, L, Tr, s^I, S^G, \varpi)$ be the transition graph of Π. For each $1 \leq i \leq k$, let $\mathcal{T}_i = (S_i, L_i, Tr_i, s_i^I, S_i^G, \varpi_i)$ be the transition graph of Π_{ϕ_i}. We need to show that α_i is an abstraction mapping as in Definition 3.

First, from Definition 15 we have

- $s_i^I = I_{\phi_i} = \alpha_i(I) = \alpha_i(s^I)$, and
- for all $s \in S^G$ we have $s \supseteq G$ and thus $\alpha_i(s) \supseteq \alpha_i(G) = G_{\phi_i}$, providing us with $\alpha_i(s) \in S_i^G$.

Now, if s is a state of Π and $a \in A$ is an action with $\mathsf{pre}(a) \subseteq s$, then $\alpha_i(s)$ is a state of Π_{ϕ_i} and $\mathsf{pre}(a_{\phi_i}) = \alpha_i(\mathsf{pre}(a)) \subseteq \alpha_i(s)$. Thus, a_{ϕ_i} is applicable in $\alpha_i(s)$, and now we show that applying a_{ϕ_i} in $\alpha_i(s)$ results in $\alpha_i(s)[\![a_{\phi_i}]\!] = \alpha_i(s[\![a]\!])$.

1. For the effect variables $v \in \mathcal{V}(\mathsf{eff}(a)) = \mathcal{V}(\mathsf{eff}(a_{\phi_i}))$, we have $\mathsf{eff}(a_{\phi_i}) \subseteq \alpha_i(s)[\![a_{\phi_i}]\!]$ and $\mathsf{eff}(a_{\phi_i}) = \alpha_i(\mathsf{eff}(a)) \subseteq \alpha_i(s[\![a]\!])$.

2. For all other variables $v \notin \mathcal{V}(\mathsf{eff}(a))$, we have $s[\![a]\!][v] = s[v]$ and $\alpha_i(s)[\![a_{\phi_i}]\!][v] = \alpha_i(s)[v]$, and thus

$$\alpha_i(s)[\![a_{\phi_i}]\!][v] = \alpha_i(s)[v] = \alpha_i(s[v]) = \alpha_i(s[\![a]\!][v]) = \alpha_i(s[\![a]\!])[v].$$

Next, for each $a \in A$, from Eqs. 4.7 and 4.8 we have

$$\sum_{i=1}^{k} cost_{\phi_i}(a_{\phi_i}) = \sum_{i=1}^{k} cost_i(a) \leq cost(a). \tag{4.9}$$

Now, let $s, s' \in S$ be a pair of states such that $cost(s, s') \leq \infty$, and let $\varrho = \langle a^1, \ldots, a^l \rangle$ be the sequence of labels along a shortest path from s to s' in $\mathcal{T}$. From that, $cost(s, s') = cost(\varrho) = \sum_{j=1}^{l} cost(a^j)$. The decomposition of such a path to the actions as in Definition 15 is a (not neccesarily shortest) path from $\alpha_i(s)$ to $\alpha_i(s')$ in $\mathcal{T}_i$, and thus $cost(\alpha_i(s), \alpha_i(s')) \leq \sum_{j=1}^{l} cost_i(a^j)$, providing us with

$$\sum_{i=1}^{k} cost(\alpha_i(s), \alpha_i(s')) \leq \sum_{i=1}^{k} \sum_{j=1}^{l} cost_{\phi_i}(a^j_{\phi_i}) = \sum_{j=1}^{l} \sum_{i=1}^{k} cost_{\phi_i}(a^j_{\phi_i}) \overset{(4.9)}{\leq} \sum_{j=1}^{l} cost(a^j) = cost(s, s').$$

■

Having put the notion of domain abstraction in the framework of implicit abstractions, we are now ready to connect fork decompositions and implicit abstraction heuristics. Given a $\mathcal{FJ}$-abstraction $\mathcal{A}_{\mathcal{FJ}} = \{\langle \Pi^{\mathsf{f}}_v, \alpha^{\mathsf{f}}_v \rangle, \langle \Pi^{\mathsf{i}}_v, \alpha^{\mathsf{i}}_v \rangle\}_{v \in V}$ of a planning task $\Pi = \langle V, A, I, G, cost \rangle$,

- for each $\Pi^{\mathsf{f}}_v \in \mathbf{\Pi}_{\mathcal{FJ}}$, we associate the root v of $CG(\Pi^{\mathsf{f}}_v)$ with mappings $\Phi^{\mathsf{f}}_v = \{\phi^{\mathsf{f}}_{v,1}, \ldots, \phi^{\mathsf{f}}_{v,k_v}\}$ such that $k_v = O(poly(||\Pi||))$ and all $\phi^{\mathsf{f}}_{v,i} : \mathcal{D}(v) \to \{0, 1\}$, and then abstract Π^{f}_v with $\mathcal{A}^{\mathsf{f}}_v = \{\langle \Pi^{\mathsf{f}}_{v,i}, \alpha^{\mathsf{f}}_{v,i} \rangle\}_{i=1}^{k_v}$, and
- for each $\Pi^{\mathsf{i}}_v \in \mathbf{\Pi}_{\mathcal{FJ}}$, we associate the sink v of $CG(\Pi^{\mathsf{i}}_v)$ with mappings $\Phi^{\mathsf{i}}_v = \{\phi^{\mathsf{i}}_{v,1}, \ldots, \phi^{\mathsf{i}}_{v,k'_v}\}$ such that $k'_v = O(poly(||\Pi||))$ and all $\phi^{\mathsf{i}}_{v,i} : \mathcal{D}(v) \to \{0, 1, \ldots, b_{v,i}\}$, $b_{v,i} = O(1)$, and then abstract Π^{i}_v with $\mathcal{A}^{\mathsf{i}}_v = \{\langle \Pi^{\mathsf{i}}_{v,i}, \alpha^{\mathsf{i}}_{v,i} \rangle\}_{i=1}^{k'_v}$.

From Theorem 8, Theorem 9, and the composition Theorem 7, we then immediately have

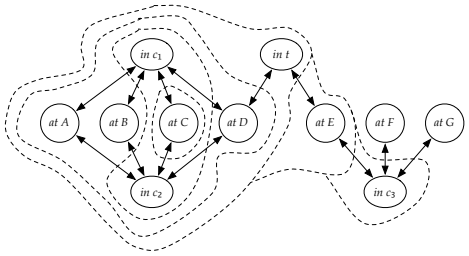
$$\mathcal{A}_{\mathcal{FJ}} = \bigcup_{v \in V} \left[\bigcup_{i=1}^{k_v} \{\langle \Pi^{\mathsf{f}}_{v,i}, \alpha^{\mathsf{f}}_{v,i} \circ \alpha^{\mathsf{f}}_v \rangle\} \cup \bigcup_{i=1}^{k'_v} \{\langle \Pi^{\mathsf{i}}_{v,i}, \alpha^{\mathsf{i}}_{v,i} \circ \alpha^{\mathsf{i}}_v \rangle\} \right] \tag{4.10}$$

being an additive implicit abstraction of Π. Hence, from Theorem 6,

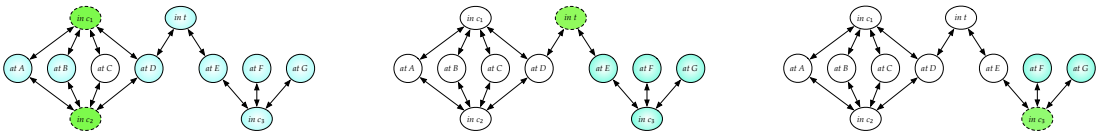
$$h^{\mathcal{FJ}} = \sum_{v \in V} \left[\sum_{i=1}^{k_v} h^*_{\Pi^{\mathsf{f}}_{v,i}} + \sum_{i=1}^{k'_v} h^*_{\Pi^{\mathsf{i}}_{v,i}} \right] \tag{4.11}$$

is an admissible estimate of h^* for Π, and, from Theorems 4 and 5, $h^{\mathcal{FJ}}$ is also computable in time $O(poly(||\Pi||))$.

This finalizes our construction of a concrete family of implicit abstraction heuristics. To illustrate the mixture of acyclic causal-graph and domain abstractions as above, we again use our running Logistics example. One bothersome question is to what extent further abstracting fork decompositions using domain abstractions affects the informativeness of the heuristic estimate. Though generally a degradation here is unavoidable, below we show that the answer to this question can sometimes be somewhat surprising.



(a)



$\mathcal{D}(p_1)$ in $\Pi^{\mathsf{f}}_{p_1,1}$ $\mathcal{D}(p_1)$ in $\Pi^{\mathsf{f}}_{p_1,2}$ $\mathcal{D}(p_1)$ in $\Pi^{\mathsf{f}}_{p_1,3}$

(b)

Figure 4.3: Domain abstractions for $\mathcal{D}(p_1)$. (a) Binary-valued domain abstractions: the values inside and outside each dashed contour are mapped to 0 and 1, respectively. (b) Ternary-valued domain abstractions: values that are mapped to the same abstract value are shown as nodes with the same color and borderline.

To begin with an extreme setting, let all the domain abstractions for roots of forks and sinks of inverted forks be to *binary-valued* domains. Among multiple options for choosing the mapping sets $\{\Phi^{\mathsf{f}}_v\}$ and $\{\Phi^{\mathsf{i}}_v\}$, here we use a simple choice of distinguishing between different values of each variable v on the basis of their cost from $I[v]$ in $DTG(v, \Pi)$. Specifically, for each $v \in V$, we set $\Phi^{\mathsf{f}}_v = \Phi^{\mathsf{i}}_v$, and, for each value $\vartheta \in \mathcal{D}(v)$ and each $1 \leq i \leq \max_{\vartheta' \in \mathcal{D}(v)} d(I[v], \vartheta')$,

$$\phi^{\mathsf{f}}_{v,i}(\vartheta) = \phi^{\mathsf{i}}_{v,i}(\vartheta) = \begin{cases} 0, & d(I[v], \vartheta) < i \\ 1, & \text{otherwise} \end{cases} \tag{4.12}$$

For example, the problem $\Pi^{\mathsf{f}}_{c_1}$ is decomposed (see the domain transition graph of c_1 on the left in Figure 2.1c) into two problems, $\Pi^{\mathsf{f}}_{c_1,1}$ and $\Pi^{\mathsf{f}}_{c_1,2}$, with the binary abstract domains of c_1 corresponding to the partitions $\{\{A\}, \{B, C, D\}\}$ and $\{\{A, D\}, \{B, C\}\}$ of $\mathcal{D}(c_1)$, respectively. As yet another example, the problem $\Pi^{\mathsf{i}}_{p_1}$ is decomposed (see the domain transition graph of p_1 in Figure 2.1d) into six problems $\Pi^{\mathsf{i}}_{p_1,1}, \ldots, \Pi^{\mathsf{i}}_{p_1,6}$ along the abstractions of $\mathcal{D}(p_1)$ depicted in Figure 4.3a. Now, given the $\mathcal{FI}$-decomposition of Π and mappings $\{\Phi^{\mathsf{f}}_v, \Phi^{\mathsf{i}}_v\}_{v \in V}$ as above, consider the problem $\Pi^{\mathsf{i}}_{p_1,1}$, obtained from abstracting Π along the inverted fork of p_1 and then abstracting $\mathcal{D}(p_1)$ using

$$\phi^{\mathsf{i}}_{p_1,1}(\vartheta) = \begin{cases} 0, & \vartheta \in \{C\} \\ 1, & \vartheta \in \{A, B, D, E, F, G, c_1, c_2, c_3, t\} \end{cases}.$$

It is not hard to verify that, from the original actions affecting p_1, we are left in $\Pi^{\mathsf{i}}_{p_1,1}$ with only actions conditioned by c_1 and c_2. If so, then no information is lost[2] if we remove from $\Pi^{\mathsf{i}}_{p_1,1}$ both variables c_3 and t, as well as the actions changing (only) these variables, and redistribute the cost of the removed actions between all other representatives of their originals in Π. The latter revision of the action cost partition can be obtained directly by replacing the cost-partitioning steps corresponding to Eqs. 4.2-4.3 and 4.7-4.8 by a single, joint action cost partitioning applied over the final additive implicit abstraction $\mathcal{A}_{\mathfrak{FJ}}$ as in Eq. 4.10 and satisfying

$$cost(a) \geq \sum_{v \in V} \left[\sum_{i=1}^{k_v} \sum_{a' \in A_{\mathcal{G}^{\mathsf{f}}_v}(a)} cost^{\mathsf{f}}_{v,i}(\phi^{\mathsf{f}}_{v,i}(a')) \quad + \sum_{i=1}^{k'_v} \sum_{a' \in A_{\mathcal{G}^{\mathsf{i}}_v}(a)} cost^{\mathsf{i}}_{v,i}(\phi^{\mathsf{i}}_{v,i}(a')) \right] . \tag{4.13}$$

In what follows, by uniform action cost partition we refer to a partition in which the cost of each action is equally split among all its nonredundant representatives in the *final* additive implicit abstraction.

Overall, computing $h^{\mathfrak{FJ}}$ as in Eq. 4.11 under our "all binary-valued domain abstractions" and such a uniform action cost partition provides us with $h^{\mathfrak{FJ}}(I) = 12\frac{7}{15}$, and knowing that the original costs are all integers we can safely adjust it to $h^{\mathfrak{FJ}}(I) = 13$. Hence, even under the most severe domain abstractions as above, the estimate of $h^{\mathfrak{FJ}}$ in our example task is not lower than that of h^2.

Let us now slightly refine our domain abstractions for the sinks of the inverted forks to be to a ternary range $\{0, 1, 2\}$. While mappings $\{\Phi^{\mathsf{f}}_v\}$ remain unchanged, $\{\Phi^{\mathsf{i}}_v\}$ are set to

$$\forall \vartheta \in \mathcal{D}(v): \quad \phi^{\mathsf{i}}_{v,i}(\vartheta) = \begin{cases} 0, d(I[v], \vartheta) < 2i-1 \\ 1, d(I[v], \vartheta) = 2i-1 \\ 2, d(I[v], \vartheta) > 2i-1 \end{cases} . \tag{4.14}$$

For example, the problem $\Pi^{\mathsf{i}}_{p_1}$ is now decomposed into $\Pi^{\mathsf{i}}_{p_1,1}, \ldots, \Pi^{\mathsf{i}}_{p_1,3}$ along the abstractions of $\mathcal{D}(p_1)$ depicted in Figure 4.3b. Applying now the same computation of $h^{\mathfrak{FJ}}$ as in Eq. 4.11 over the new set of domain abstractions gives $h^{\mathfrak{FJ}}(I) = 15\frac{1}{2}$, which, again, can be safely adjusted to $h^{\mathfrak{FJ}}(I) = 16$. Note that this value is *higher* than $h^{\mathfrak{FJ}} = 15$ obtained using the (generally intractable) "pure" fork-decomposition abstractions as in Eq. 4.5. At first view, this outcome may seem counterintuitive as the domain abstractions are applied *over* the fork decomposition, and one would expect a coarser abstraction to provide less precise estimates. This, however, is not necessarily the case when the employed action cost partition is ad hoc. For instance, domain abstraction for the sink of an inverted fork may create independence between the sink and its parent variables, and exploiting such domain-abstraction specific independence relations leads to more targeted action cost partition via Eq. 4.13.

To see why this surprising "estimate improvement" has been obtained, note that before the domain abstraction in Eq. 4.14 is applied on our example, the truck-moving actions *drive-t-from-D-to-E* and *drive-t-from-E-to-D* appear in *three* abstractions Π^{f}_t, $\Pi^{\mathsf{i}}_{p_1}$ and $\Pi^{\mathsf{i}}_{p_2}$, while after domain abstraction they appear in *five* abstractions $\Pi^{\mathsf{f}}_{t,1}$, $\Pi^{\mathsf{i}}_{p_1,1}$, $\Pi^{\mathsf{i}}_{p_1,2}$, $\Pi^{\mathsf{i}}_{p_1,3}$ and $\Pi^{\mathsf{i}}_{p_2,1}$. However, a closer look at the action sets of these five abstractions reveals that the dependencies of p_1 in $CG(\Pi^{\mathsf{i}}_{p_1,1})$ and $CG(\Pi^{\mathsf{i}}_{p_1,3})$, and of p_2 in $CG(\Pi^{\mathsf{i}}_{p_2,1})$ on t are redundant, and thus keeping representatives of *move-D-E* and *move-E-D* in the corresponding abstract tasks is entirely unnecessary. Hence, after

[2]No information is lost here because we still keep either fork or inverted fork for each variable of Π.

Domain	h^+	h^k	h^{PDB}	$h^{\text{PDB}}_{\text{add}}$	$h^{\mathcal{F}}$	$h^{\mathcal{J}}$	$h^{\mathcal{FJ}}$
GRIPPER	2/3	0	0	2/3	2/3	0	4/9
LOGISTICS	3/4	0	0	1/2	1/2	1/2	1/2
BLOCKSWORLD	1/4	0	0	0	0	0	0
MICONIC-STRIPS	6/7	0	0	1/2	5/6	1/2	1/2
SATELLITE	1/2	0	0	1/6	1/6	1/6	1/6

Table 4.1: Performance ratios of multiple heuristics in selected planning domains; ratios for h^+, h^k, h^{PDB}, $h^{\text{PDB}}_{\text{add}}$ are by Helmert and Mattmüller (2008).

all, the two truck-moving actions appear only in *two* post-domain-abstraction tasks. Moreover, in both these abstractions the truck-moving actions are fully counted, in contrast to the predomain-abstraction tasks where the portion of the cost of these actions allocated to $\Pi^{\text{i}}_{p_2}$ simply gets lost.

4.4 Accuracy of Fork-Decomposition Heuristics

Empirical evaluation on a concrete set of benchmark tasks is a standard and important methodology for assessing the effectiveness of heuristic estimates: it allows us to study the tradeoff between the accuracy of the heuristics and the complexity of computing them. However, as rightfully noted by Helmert and Mattmüller (2008), such evaluations almost never lead to absolute statements of the type "Heuristic h is well-suited for solving problems from benchmark suite X," but only to relative statements of the type "Heuristic h expands fewer nodes than heuristic h' on benchmark suite X." Moreover, one would probably like to obtain formal evidence of the effectiveness of a heuristic before proceeding with its implementation, especially for very complicated heuristic procedures such as those underlying the proofs of Theorems 11 and 12.

4.4.1 Asymptotic Performance Analysis

Our formal analysis of the effectiveness of the fork-decomposition heuristics using the methodology suggested and exploited by Helmert and Mattmüller (2008) was motivated primarily by this desire for formal evidence.

Given a planning domain $\mathcal{D}$ and heuristic h, Helmert and Mattmüller (2008) consider the *asymptotic performance ratio* of h in $\mathcal{D}$. The goal is to find a value $\alpha(h, \mathcal{D}) \in [0, 1]$ such that

(1) for all states s in all problems $\Pi \in \mathcal{D}$, $h(s) \geq \alpha(h, \mathcal{D}) \cdot h^*(s) + o(h^*(s))$, and

(2) there is a family of problems $\{\Pi_n\}_{n \in \mathbb{N}} \subseteq \mathcal{D}$ and solvable, non-goal states $\{s_n\}_{n \in \mathbb{N}}$ such that $s_n \in \Pi_n$, $\lim_{n \to \infty} h^*(s_n) = \infty$, and $h(s_n) \leq \alpha(h, \mathcal{D}) \cdot h^*(s_n) + o(h^*(s_n))$.

In other words, h is *never* worse than $\alpha(h, \mathcal{D}) \cdot h^*$ (plus a sublinear term), and it can become as bad as $\alpha(h, \mathcal{D}) \cdot h^*$ (plus a sublinear term) for arbitrarily large inputs; note that both the existence and uniqueness of $\alpha(h, \mathcal{D})$ are guaranteed for any h and $\mathcal{D}$.

Helmert and Mattmüller (2008) study the asymptotic performance ratio of some standard admissible heuristics on a set of well-known benchmark domains from the first four IPCs. Their results for GRIPPER, LOGISTICS, BLOCKSWORLD, MICONIC, and SATELLITE are shown in the first four columns of Table 4.1.

- The h^+ estimate corresponds to the optimal cost of solving the well-known "delete relaxation" of the original planning task, which is generally NP-hard to compute (Bylander, 1994).

- The h^k, $k \in \mathbb{N}^+$, family of heuristics is based on a relaxation where the cost of achieving a partial assignment is approximated by the highest cost of achieving its sub-assignment of size k (Haslum & Geffner, 2000); computing h^k is exponential only in k.

- The h^{PDB} and $h^{\text{PDB}}_{\text{add}}$ heuristics are regular (maximized over) and additive pattern database heuristics where the size of each pattern is assumed to be $O(\log(n))$ where $n = |V|$, and, importantly, the choice of the patterns is assumed to be optimal.

These results provide us with a baseline for evaluating our fork-decomposition heuristics $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$. First, however, Theorem 10 shows that these three heuristics are worth analyzing because each alone can be strictly more informative than the other two, depending on the planning task and/or the state being evaluated.[3]

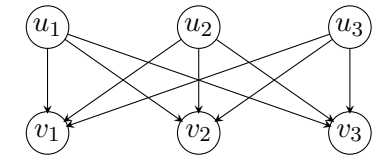
Theorem 10 (Undominance) *Under uniform action cost partition, none of the heuristic functions $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$ dominates another.*

Proof: The proof is by example of two tasks, Π^1 and Π^2, which illustrate the following two cases: $h^{\mathcal{F}}(I) > h^{\mathcal{FJ}}(I) > h^{\mathcal{J}}(I)$ and $h^{\mathcal{F}}(I) < h^{\mathcal{FJ}}(I) < h^{\mathcal{J}}(I)$. These two tasks are defined over the same set of binary-valued variables $V = \{v_1, v_2, v_3, u_1, u_2, u_3\}$, have the same initial state $I = \{v_1{=}0, v_2{=}0, v_3{=}0, u_1{=}0, u_2{=}0, u_3{=}0\}$, and have the same goal $G = \{v_1{=}1, v_2{=}1, v_3{=}1\}$. The difference between Π^1 and Π^2 is in the action sets, listed in Figure 4.4c-d, with all the actions being unit-cost actions. The two tasks induce identical causal graphs, depicted in Figure 4.4a. Hence, the collections of v-forks and v-iforks of both tasks are also identical; these are depicted in Figure 4.4b. The fractional costs of the tasks' action representatives in the corresponding abstract problems are given in Figure 4.4c-d.

Figure 4.5 shows the optimal plans for all the abstract problems in $\mathcal{F}$-decompositions $\mathbf{\Pi}^1_{\mathcal{F}} = \{\Pi^1_{\mathcal{G}^{\text{f}}_{u_1}}, \Pi^1_{\mathcal{G}^{\text{f}}_{u_2}}, \Pi^1_{\mathcal{G}^{\text{f}}_{u_3}}\}$ and $\mathbf{\Pi}^2_{\mathcal{F}} = \{\Pi^2_{\mathcal{G}^{\text{f}}_{u_1}}, \Pi^2_{\mathcal{G}^{\text{f}}_{u_2}}, \Pi^2_{\mathcal{G}^{\text{f}}_{u_3}}\}$, $\mathcal{J}$-decompositions $\mathbf{\Pi}^1_{\mathcal{J}} = \{\Pi^1_{\mathcal{G}^{\text{i}}_{v_1}}, \Pi^1_{\mathcal{G}^{\text{i}}_{v_2}}, \Pi^1_{\mathcal{G}^{\text{i}}_{v_3}}\}$ and $\mathbf{\Pi}^2_{\mathcal{J}} = \{\Pi^2_{\mathcal{G}^{\text{i}}_{v_1}}, \Pi^2_{\mathcal{G}^{\text{i}}_{v_2}}, \Pi^2_{\mathcal{G}^{\text{i}}_{v_3}}\}$, and $\mathcal{FJ}$-decompositions $\mathbf{\Pi}^1_{\mathcal{FJ}} = \mathbf{\Pi}^1_{\mathcal{F}} \cup \mathbf{\Pi}^1_{\mathcal{J}}$ and $\mathbf{\Pi}^2_{\mathcal{FJ}} = \mathbf{\Pi}^2_{\mathcal{F}} \cup \mathbf{\Pi}^2_{\mathcal{J}}$. The last column in both tables captures the estimates of the three heuristics for the initial states of Π^1 and Π^2, respectively. Together, these two cases show that none of the fork-decomposition heuristic functions $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$ dominates any other, and, since all the variables above are binary-valued, the claim holds in conjunction with *arbitrary* variable domain abstractions. ∎

One conclusion from Theorem 10 is that it is worth studying the asymptotic performance ratios for all three heuristics. The last three columns of Table 4.1 present our results for $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$ for the Gripper, Logistics, Blocksworld, Miconic, and Satellite domains. We also studied the performance ratios of $\max\{h^{\mathcal{F}}, h^{\mathcal{J}}, h^{\mathcal{FJ}}\}$, and in these five domains they appear to be identical to those of $h^{\mathcal{F}}$. (Note that "ratio of max" should not necessarily be identical to "max of ratios," and thus this analysis is worthwhile.) Taking a conservative position, the performance ratios for the fork-decomposition heuristics in Table 4.1 are "worst-case" in the sense that

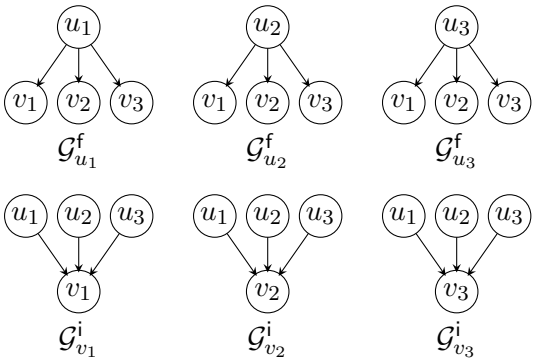
[3]Theorem 10 is formulated and proven under the uniform action cost partition that we use throughout this chapter, including the experiments. For per-step optimal action cost partitions (Katz & Domshlak, 2008b), it is trivial to show that $h^{\mathcal{FJ}}$ dominates both $h^{\mathcal{F}}$ and $h^{\mathcal{J}}$ for all planning tasks.



(a)

	A_1	$\mathbf{\Pi}^1_{\mathcal{F}}$	$\mathbf{\Pi}^1_{\mathcal{I}}$	$\mathbf{\Pi}^1_{\mathcal{FI}}$
a_1	$\langle\{v_1=0,u_1=0,u_2=0,u_3=0\},\{v_1=1\}\rangle$	1/3	1	1/4
a_2	$\langle\{v_2=0,u_1=1,u_2=0,u_3=1\},\{v_2=1\}\rangle$	1/3	1	1/4
a_3	$\langle\{v_3=0,u_1=1,u_2=1,u_3=0\},\{v_3=1\}\rangle$	1/3	1	1/4
a_4	$\langle\{u_1=0\},\{u_1=1\}\rangle$	1	1/3	1/4
a_5	$\langle\{u_1=1\},\{u_1=0\}\rangle$	1	1/3	1/4
a_6	$\langle\{u_2=0\},\{u_2=1\}\rangle$	1	1/3	1/4
a_7	$\langle\{u_2=1\},\{u_2=0\}\rangle$	1	1/3	1/4
a_8	$\langle\{u_3=0\},\{u_3=1\}\rangle$	1	1/3	1/4
a_9	$\langle\{u_3=1\},\{u_3=0\}\rangle$	1	1/3	1/4

(c)



(b)

	A_2	$\mathbf{\Pi}^2_{\mathcal{F}}$	$\mathbf{\Pi}^2_{\mathcal{I}}$	$\mathbf{\Pi}^2_{\mathcal{FI}}$
a_1	$\langle\{v_1=0,u_1=1\},\{v_1=1\}\rangle$	1/3	1	1/4
a_2	$\langle\{v_1=0,u_2=1\},\{v_1=1\}\rangle$	1/3	1	1/4
a_3	$\langle\{v_1=0,u_3=1\},\{v_1=1\}\rangle$	1/3	1	1/4
a_4	$\langle\{v_2=0,u_1=1\},\{v_2=1\}\rangle$	1/3	1	1/4
a_5	$\langle\{v_2=0,u_2=1\},\{v_2=1\}\rangle$	1/3	1	1/4
a_6	$\langle\{v_2=0,u_3=1\},\{v_2=1\}\rangle$	1/3	1	1/4
a_7	$\langle\{v_3=0,u_1=1\},\{v_3=1\}\rangle$	1/3	1	1/4
a_8	$\langle\{v_3=0,u_2=1\},\{v_3=1\}\rangle$	1/3	1	1/4
a_9	$\langle\{v_3=0,u_3=1\},\{v_3=1\}\rangle$	1/3	1	1/4
a_{10}	$\langle\{u_1=0\},\{u_1=1\}\rangle$	1	1/3	1/4
a_{11}	$\langle\{u_2=0\},\{u_2=1\}\rangle$	1	1/3	1/4
a_{12}	$\langle\{u_3=0\},\{u_3=1\}\rangle$	1	1/3	1/4

(d)

Figure 4.4: Illustrations for the proof of Theorem 10: (a) causal graphs of Π^1 and Π^2, (b) fork and inverted fork subgraphs of the (same) causal graph of Π^1 and Π^2, and the action sets of (c) Π^1 and (d) Π^2, as well as the costs of the action representatives in each abstract problem along these subgraphs. Considering for example the first row of table (c), the action a_1 in Π^1 has a single representative in each of the three fork abstractions, as well as a representative in the inverted-fork abstraction $\Pi^1_{\mathcal{G}^{\mathsf{i}}_{v_1}}$. Hence, the cost of each of its representatives in $\mathcal{F}$-decomposition is 1/3, while the cost of its sole representative in $\mathcal{I}$-decomposition is 1.

(i) here we neither optimize the action cost partition (setting it to uniform as in the rest of this chapter) nor eliminate clearly redundant abstractions, and

(ii) we use domain abstractions to (up to) ternary-valued abstract domains only.

The domains of the fork roots are all abstracted using the "leave-one-out" binary-valued domain decompositions as in Eq. 4.24 while the domains of the inverted-fork sinks are all abstracted using the "distance-from-initial-value" ternary-valued domain decompositions as in Eq. 4.14.

Overall, the results for fork-decomposition heuristics in Table 4.1 are gratifying. First, note that the performance ratios for h^k and h^{PDB} are all 0. This is because every subgoal set of size k (for h^k) and size $\log(n)$ (for h^{PDB}) can be reached in the number of steps that only depends on k (respectively, $\log(n)$), and not n, while $h^*(s_n)$ grows linearly in n in all the five domains. This leaves us with $h^{\text{PDB}}_{\text{add}}$ being the only state-of-the-art (tractable and) admissible heuristic to compare with. Table 4.1 shows that the asymptotic performance ratio of $h^{\mathcal{F}}$ heuristic is at least as good as that of $h^{\text{PDB}}_{\text{add}}$ in *all* five domains, while $h^{\mathcal{F}}$ is superior to $h^{\text{PDB}}_{\text{add}}$ in Miconic, getting here quite close to h^+. When comparing $h^{\text{PDB}}_{\text{add}}$ and fork-decomposition heuristics, it is crucial to recall that the ratios devised by Helmert and Mattmüller for $h^{\text{PDB}}_{\text{add}}$ are with respect to optimal, manually-selected

h	task	optimal plan	cost	$h(I)$
$h^{\mathcal{F}}$	$\Pi^1_{\mathcal{G}^{\mathrm{f}}_{u_1}}$	$\langle a_1 \cdot a_4 \cdot a_2 \cdot a_3 \rangle$	2	6
	$\Pi^1_{\mathcal{G}^{\mathrm{f}}_{u_2}}$	$\langle a_1 \cdot a_2 \cdot a_6 \cdot a_3 \rangle$	2	
	$\Pi^1_{\mathcal{G}^{\mathrm{f}}_{u_3}}$	$\langle a_1 \cdot a_3 \cdot a_8 \cdot a_2 \rangle$	2	
$h^{\mathcal{J}}$	$\Pi^1_{\mathcal{G}^{\mathrm{i}}_{v_1}}$	$\langle a_1 \rangle$	1	$4\frac{1}{3}$
	$\Pi^1_{\mathcal{G}^{\mathrm{i}}_{v_2}}$	$\langle a_4 \cdot a_8 \cdot a_2 \rangle$	5/3	
	$\Pi^1_{\mathcal{G}^{\mathrm{i}}_{v_3}}$	$\langle a_4 \cdot a_6 \cdot a_3 \rangle$	5/3	
$h^{\mathcal{FJ}}$	$\Pi^1_{\mathcal{G}^{\mathrm{f}}_{u_1}}$	$\langle a_1 \cdot a_4 \cdot a_2 \cdot a_3 \rangle$	1	$4\frac{3}{4}$
	$\Pi^1_{\mathcal{G}^{\mathrm{f}}_{u_2}}$	$\langle a_1 \cdot a_2 \cdot a_6 \cdot a_3 \rangle$	1	
	$\Pi^1_{\mathcal{G}^{\mathrm{f}}_{u_3}}$	$\langle a_1 \cdot a_3 \cdot a_8 \cdot a_2 \rangle$	1	
	$\Pi^1_{\mathcal{G}^{\mathrm{i}}_{v_1}}$	$\langle a_1 \rangle$	1/4	
	$\Pi^1_{\mathcal{G}^{\mathrm{i}}_{v_2}}$	$\langle a_4 \cdot a_8 \cdot a_2 \rangle$	3/4	
	$\Pi^1_{\mathcal{G}^{\mathrm{i}}_{v_3}}$	$\langle a_4 \cdot a_6 \cdot a_3 \rangle$	3/4	

(a)

h	task	optimal plan	cost	$h(I)$
$h^{\mathcal{F}}$	$\Pi^2_{\mathcal{G}^{\mathrm{f}}_{u_1}}$	$\langle a_2 \cdot a_5 \cdot a_8 \rangle$	1	3
	$\Pi^2_{\mathcal{G}^{\mathrm{f}}_{u_2}}$	$\langle a_1 \cdot a_4 \cdot a_7 \rangle$	1	
	$\Pi^2_{\mathcal{G}^{\mathrm{f}}_{u_3}}$	$\langle a_1 \cdot a_4 \cdot a_7 \rangle$	1	
$h^{\mathcal{J}}$	$\Pi^2_{\mathcal{G}^{\mathrm{i}}_{v_1}}$	$\langle a_{10} \cdot a_1 \rangle$	4/3	4
	$\Pi^2_{\mathcal{G}^{\mathrm{i}}_{v_2}}$	$\langle a_{10} \cdot a_4 \rangle$	4/3	
	$\Pi^2_{\mathcal{G}^{\mathrm{i}}_{v_3}}$	$\langle a_{10} \cdot a_7 \rangle$	4/3	
$h^{\mathcal{FJ}}$	$\Pi^2_{\mathcal{G}^{\mathrm{f}}_{u_1}}$	$\langle a_2 \cdot a_5 \cdot a_8 \rangle$	3/4	15/4
	$\Pi^2_{\mathcal{G}^{\mathrm{f}}_{u_2}}$	$\langle a_1 \cdot a_4 \cdot a_7 \rangle$	3/4	
	$\Pi^2_{\mathcal{G}^{\mathrm{f}}_{u_3}}$	$\langle a_1 \cdot a_4 \cdot a_7 \rangle$	3/4	
	$\Pi^2_{\mathcal{G}^{\mathrm{i}}_{v_1}}$	$\langle a_{10} \cdot a_1 \rangle$	1/2	
	$\Pi^2_{\mathcal{G}^{\mathrm{i}}_{v_2}}$	$\langle a_{10} \cdot a_4 \rangle$	1/2	
	$\Pi^2_{\mathcal{G}^{\mathrm{i}}_{v_3}}$	$\langle a_{10} \cdot a_7 \rangle$	1/2	

(b)

Figure 4.5: Illustrations for the proof of Theorem 10: Optimal plans for all the abstract problems of (a) Π^1, where we have $h^{\mathcal{F}}(I) > h^{\mathcal{FJ}}(I) > h^{\mathcal{J}}(I)$, and (b) Π^2, where we have $h^{\mathcal{F}}(I) < h^{\mathcal{FJ}}(I) < h^{\mathcal{J}}(I)$.

set of patterns. By contrast, the selection of variable subsets for fork-decomposition heuristics is completely nonparametric, and thus requires no tuning of the abstraction-selection process.

In the rest of the section we prove these asymptotic performance ratios of $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$ in Table 4.1 for the five domains. We begin with a very brief outline of how the results are obtained. *Some* familiarity with the domains is assumed. Next, each domain is addressed in detail: we provide an informal domain description as well as its SAS$^+$ representation, and then prove lower and upper bounds on the ratios for all three heuristics.

Gripper Assuming $n > 0$ balls should be moved from one room to another, all three heuristics $h^{\mathcal{F}}, h^{\mathcal{J}}, h^{\mathcal{FJ}}$ account for all the required pickup and drop actions, and only for $O(1)$-portion of move actions. However, the former actions are responsible for 2/3 of the optimal-plan length (= cost). Now, with the basic uniform action-cost partition, $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$ account for the whole, $O(1/n)$, and 2/3 of the total pickup/drop actions cost, respectively, providing the ratios in Table 4.1.[4]

Logistics An optimal plan contains at least as many load/unload actions as move actions, and all three heuristics $h^{\mathcal{F}}, h^{\mathcal{J}}, h^{\mathcal{FJ}}$ fully account for the former, providing a lower bound of 1/2. An instance on which all three heuristics achieve exactly 1/2 consists of two trucks t_1, t_2, no airplanes, one city, and n packages such that the initial and goal locations of all the packages and trucks are all pair-wise different.

Blocksworld Arguments similar to those of Helmert and Mattmüller (2008) for $h^{\mathrm{PDB}}_{\mathrm{add}}$.

Miconic All three heuristics fully account for all the loads/unload actions. In addition, $h^{\mathcal{F}}$ accounts for the full cost of all the move actions to the passengers' initial locations, and for half of the cost of all the other move actions. This provides us with lower bounds of 1/2 and 5/6,

[4]We note that a very slight modification of the uniform action-cost partition results in a ratio of 2/3 for all three heuristics. Such optimizations, however, are outside of our scope here.

respectively. Tightness of 1/2 for $h^{\mathfrak{I}}$ and $h^{\mathfrak{FJ}}$ is shown on a task consisting of n passengers, $2n+1$ floors, and all the initial and goal locations being pair-wise different. Tightness of 5/6 for $h^{\mathfrak{F}}$ is shown on a task consisting of n passengers, $n+1$ floors, the elevator and all the passengers are initially at floor $n+1$, and each passenger i wishes to get to floor i.

Satellite The length of an optimal plan for a problem with n images to be taken and k satellites to be moved to some end-positions is $\leq 6n+k$. All three heuristics fully account for all the image-taking actions and one satellite-moving action per satellite as above, providing a lower bound of $\frac{1}{6}$. Tightness of 1/6 for all three heuristics is shown on a task as follows: Two satellites with instruments $\{i\}_{i=1}^{l}$ and $\{i\}_{i=l+1}^{2l}$, respectively, where $l = n - \sqrt{n}$. Each pair of instruments $\{i, l+i\}$ can take images in modes $\{m_0, m_i\}$. There is a set of directions $\{d_j\}_{j=0}^{n}$ and a set of image objectives $\{o_i\}_{i=1}^{n}$ such that, for $1 \leq i \leq l$, $o_i = (d_0, m_i)$ and, for $l < i \leq n$, $o_i = (d_i, m_0)$. Finally, the calibration direction for each pair of instruments $\{i, l+i\}$ is d_i.

Gripper

The domain consists of one robot *robot* with two arms $Arms = \{right, left\}$, two rooms $Rooms = \{r1, r2\}$, and a set $Balls$ of n balls. The robot can pick up a ball with an arm $arm \in Arms$ if arm is empty, release a ball $b \in Balls$ from the arm arm if arm currently holds b, and move from one room to another. All balls and the robot are initially in room $r1$, both arms are empty, and the goal is to move all the balls to room $r2$. A natural description of this planning task in SAS$^+$ is as follows.

- Variables $V = \{robot\} \bigcup Arms \bigcup Balls$ with domains

$$\begin{aligned} \mathcal{D}(robot) &= Rooms \\ \mathcal{D}(left) = \mathcal{D}(right) &= Balls \cup \{\mathsf{empty}\} \\ \forall b \in Balls:\ \mathcal{D}(b) &= Rooms \cup \{\mathsf{robot}\}. \end{aligned}$$

- Initial state $I = \{b = \mathsf{r1} \mid b \in Balls\} \cup \{robot = \mathsf{r1}, right = \mathsf{empty}, left = \mathsf{empty}\}$.
- Goal $G = \{b = \mathsf{r2} \mid b \in Balls\}$.
- Actions

$$\begin{aligned} A = &\{Move(r, r') \mid \{r, r'\} \subseteq Rooms\} \bigcup \\ &\{Pickup(b, arm, r), Drop(b, arm, r) \mid b \in Balls, arm \in Arms, r \in Rooms\}, \end{aligned}$$

  where

  - move robot: $Move(r, r') = \langle \{robot = \mathsf{r}\}, \{robot = \mathsf{r}'\} \rangle$,
  - pickup ball:
    $Pickup(b, arm, r) = \langle \{b = \mathsf{r}, arm = \mathsf{empty}, robot = \mathsf{r}\}, \{b = \mathsf{robot}, arm = \mathsf{b}\} \rangle$, and
  - drop ball: $Drop(b, arm, r) = \langle \{b = \mathsf{robot}, arm = \mathsf{b}, robot = \mathsf{r}\}, \{b = \mathsf{r}, arm = \mathsf{empty}\} \rangle$.

  The (parametric in n) causal graph of this task is depicted in Figure 4.6a.

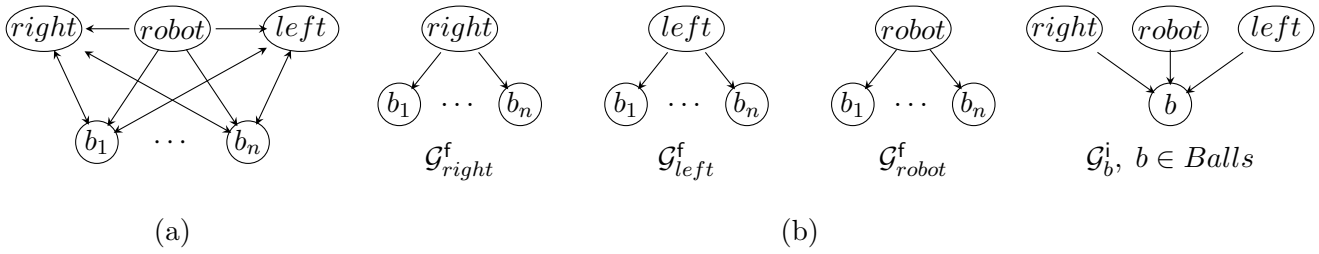


Figure 4.6: GRIPPER's (a) causal graph and (b) the corresponding collection of v-forks and v-iforks

Action	Π^{f}_{robot}	$\Pi^{\mathsf{f}}_{arm,\mathsf{empty}}$	$\Pi^{\mathsf{f}}_{arm,b}$	$\Pi^{\mathsf{f}}_{arm,b'}$	$\Pi^{\mathsf{f}}_{arm',\vartheta}$	Π^{i}_{b}	$\Pi^{\mathsf{i}}_{b'}$	$\boldsymbol{\Pi}_{\mathcal{F}}$	$\boldsymbol{\Pi}_{\mathcal{I}}$	$\boldsymbol{\Pi}_{\mathcal{FI}}$
$Move(r, r')$	1	0	0	0	0	1	1	1	$\frac{1}{n}$	$\frac{1}{n+1}$
$Pickup(b, arm, r)$	1	2	2	1	1	2	1	$\frac{1}{2n+5}$	$\frac{1}{n+1}$	$\frac{1}{3n+6}$
$Drop(b, arm, r)$	1	2	2	1	1	2	1	$\frac{1}{2n+5}$	$\frac{1}{n+1}$	$\frac{1}{3n+6}$

Table 4.2: Number of representatives for each original GRIPPER action in each abstract task, as well as the partition of the action costs between these representatives

Π^{f}_{robot}	$Pickup(b, right, \mathsf{r1}) = \langle\{robot = \mathsf{r1}, b = \mathsf{r1}\}, \{b = \mathsf{robot}\}\rangle$
$\Pi^{\mathsf{f}}_{right,\mathsf{empty}}$	$Pickup(b, right, \mathsf{r1})^1 = \langle\{right = \mathsf{empty}\}, \{right = \mathsf{b}\}\rangle$, $Pickup(b, right, \mathsf{r1})^2 = \langle\{right = \mathsf{b}, b = \mathsf{r1}\}, \{b = \mathsf{robot}\}\rangle$
$\Pi^{\mathsf{f}}_{right,b}$	$Pickup(b, right, \mathsf{r1})^1 = \langle\{right = \mathsf{empty}\}, \{right = \mathsf{b}\}\rangle$, $Pickup(b, right, \mathsf{r1})^2 = \langle\{right = \mathsf{b}, b = \mathsf{r1}\}, \{b = \mathsf{robot}\}\rangle$
$\Pi^{\mathsf{f}}_{right,b'}$	$Pickup(b, right, \mathsf{r1}) = \langle\{right = \mathsf{b}, b = \mathsf{r1}\}, \{b = \mathsf{robot}\}\rangle$
$\Pi^{\mathsf{f}}_{left,\vartheta}$	$Pickup(b, right, \mathsf{r1}) = \langle\{right = \mathsf{b}, b = \mathsf{r1}\}, \{b = \mathsf{robot}\}\rangle$
Π^{i}_{b}	$Pickup(b, right, \mathsf{r1})^1 = \langle\{right = \mathsf{empty}\}, \{right = b\}\rangle$, $Pickup(b, right, \mathsf{r1})^2 = \langle\{right = \mathsf{b}, robot = \mathsf{r1}, b = \mathsf{r1}\}, \{b = \mathsf{robot}\}\rangle$
$\Pi^{\mathsf{i}}_{b'}$	$Pickup(b, right, \mathsf{r1}) = \langle\{right = \mathsf{empty}\}, \{right = \mathsf{b}\}\rangle$

Table 4.3: The sets of representatives of the original action $Pickup(b, right, \mathsf{r1})$ in the abstract tasks

Fork Decomposition Since the variables $robot$, $right$, and $left$ have no goal value, the collection of v-forks and v-iforks is as in Figure 4.6b. The domains of inverted fork sinks are ternary valued. The domains of fork roots are abstracted as in Eq. 4.24 ("leave one out"), p. 69, and thus

$$\begin{aligned}
\boldsymbol{\Pi}_{\mathcal{F}} &= \{\Pi^{\mathsf{f}}_{robot}\} \cup \{\Pi^{\mathsf{f}}_{right,\vartheta}, \Pi^{\mathsf{f}}_{left,\vartheta} \mid \vartheta \in \{\mathsf{empty}\} \cup Balls\},\\
\boldsymbol{\Pi}_{\mathcal{I}} &= \{\Pi^{\mathsf{i}}_{b} \mid b \in Balls\},\\
\boldsymbol{\Pi}_{\mathcal{FI}} &= \{\Pi^{\mathsf{f}}_{robot}\} \cup \{\Pi^{\mathsf{f}}_{right,\vartheta}, \Pi^{\mathsf{f}}_{left,\vartheta} \mid \vartheta \in \{\mathsf{empty}\} \cup Balls\} \cup \{\Pi^{\mathsf{i}}_{b} \mid b \in Balls\}.
\end{aligned}$$

For each original action, the number of its representatives in each abstract task, as well as the cost assigned to each such representative, are listed in Table 4.2. Table 4.3 illustrates derivation of these numbers via decomposition of an example action $Pickup(b, right, \mathsf{r1})$ in each of the fork decomposition abstractions. That action has one nonredundant representative in Π^{f}_{robot}, two such representatives in each of $\Pi^{\mathsf{f}}_{right,\mathsf{empty}}$ and $\Pi^{\mathsf{f}}_{right,b}$, one representative in each $\Pi^{\mathsf{f}}_{right,b'}$ for $b' \in$

$Balls \setminus \{b\}$, one representative in each $\Pi^{\mathsf{f}}_{left,\vartheta}$ for $\vartheta \in Balls \cup \{\mathsf{empty}\}$, two representatives in Π^{i}_{b}, and one representative in each $\Pi^{\mathsf{i}}_{b'}$ for $b' \in Balls \setminus \{b\}$. This results in cost $\frac{1}{2n+5}$ for each representative in $\mathbf{\Pi}_{\mathcal{F}}$, $\frac{1}{n+1}$ for each representative in $\mathbf{\Pi}_{\mathcal{J}}$, and $\frac{1}{3n+6}$ for each representative in $\mathbf{\Pi}_{\mathcal{FJ}}$.

Given that, the optimal plans for the abstract tasks are as follows.

h	task	optimal plan	cost	#	$h(I)$
$h^{\mathcal{F}}$	Π^{f}_{robot}	$\langle Pickup(b_1, right, \mathsf{r1}) \cdot \ldots \cdot Pickup(b_n, right, \mathsf{r1}) \cdot$ $\cdot Move(\mathsf{r1}, \mathsf{r2}) \cdot Drop(b_1, right, \mathsf{r2}) \cdot \ldots \cdot Drop(b_n, right, \mathsf{r2}) \rangle$	$\frac{4n+5}{2n+5}$	1	$2n - \frac{2n-5}{2n+5}$
	$\Pi^{\mathsf{f}}_{right,\vartheta}$	$\langle Pickup(b_1, left, \mathsf{r1}) \cdot \ldots \cdot Pickup(b_n, left, \mathsf{r1}) \cdot$ $\cdot Drop(b_1, left, \mathsf{r2}) \cdot \ldots \cdot Drop(b_n, left, \mathsf{r2}) \rangle$	$\frac{2n}{2n+5}$	$n+1$	
	$\Pi^{\mathsf{f}}_{left,\vartheta}$	$\langle Pickup(b_1, right, \mathsf{r1}) \cdot \ldots \cdot Pickup(b_n, right, \mathsf{r1}) \cdot$ $\cdot Drop(b_1, right, \mathsf{r2}) \cdot \ldots \cdot Drop(b_n, right, \mathsf{r2}) \rangle$	$\frac{2n}{2n+5}$	$n+1$	
$h^{\mathcal{J}}$	Π^{i}_{b}	$\langle Pickup(b, right, \mathsf{r1})^1 \cdot Pickup(b, right, \mathsf{r1})^2 \cdot Move(\mathsf{r1}, \mathsf{r2}) \cdot Drop(b, left, \mathsf{r2})^2 \rangle$	$\frac{3}{n+1} + \frac{1}{n}$	n	$\frac{4n+1}{n+1}$
$h^{\mathcal{FJ}}$	Π^{f}_{robot}	$\langle Pickup(b_1, right, \mathsf{r1}) \cdot \ldots \cdot Pickup(b_n, right, \mathsf{r1}) \cdot Move(\mathsf{r1}, \mathsf{r2}) \cdot$ $\cdot Drop(b_1, right, \mathsf{r2}) \cdot \ldots \cdot Drop(b_n, right, \mathsf{r2}) \rangle$	$\frac{2n}{3n+6} + \frac{1}{n+1}$	1	$\frac{4n}{3} + \frac{4n+6}{3n+6}$
	$\Pi^{\mathsf{f}}_{right,\vartheta}$	$\langle Pickup(b_1, left, \mathsf{r1}) \cdot \ldots \cdot Pickup(b_n, left, \mathsf{r1}) \cdot$ $\cdot Drop(b_1, left, \mathsf{r2}) \cdot \ldots \cdot Drop(b_n, left, \mathsf{r2}) \rangle$	$\frac{2n}{3n+6}$	$n+1$	
	$\Pi^{\mathsf{f}}_{left,\vartheta}$	$\langle Pickup(b_1, right, \mathsf{r1}) \cdot \ldots \cdot Pickup(b_n, right, \mathsf{r1}) \cdot$ $\cdot Drop(b_1, right, \mathsf{r2}) \cdot \ldots \cdot Drop(b_n, right, \mathsf{r2}) \rangle$	$\frac{2n}{3n+6}$	$n+1$	
	Π^{i}_{b}	$\langle Pickup(b, right, \mathsf{r1})^1 \cdot Pickup(b, right, \mathsf{r1})^2 \cdot Move(\mathsf{r1}, \mathsf{r2}) \cdot Drop(b, left, \mathsf{r2})^2 \rangle$	$\frac{3}{3n+6} + \frac{1}{n+1}$	n	

Assuming $n > 0$ balls should be moved from one room to another, the cost of the optimal plan for the original task is $3n - 1$ when n is even, and $3n$ when n is odd. Therefore, the asymptotic performance ratios for the heuristics $h^{\mathcal{F}}, h^{\mathcal{J}}, h^{\mathcal{FJ}}$ on GRIPPER are 2/3, 0, and 4/9, respectively.

Logistics

Each LOGISTICS task consists of some k cities, x airplanes, y trucks and n packages. Each city i is associated with a set $L_i = \{l_i^1 \ldots, l_i^{\alpha_i}\}$ of locations within that city; the union of the locations of all the cities is denoted by $L = \bigcup_{i=1}^{k} L_i$. In addition, precisely one location in each city is an airport, and the set of airports is $L^A = \{l_1^1 \ldots, l_k^1\} \subseteq L$. Each truck can move only within the city in which it is located, and airplanes can fly between airports. The airplanes are denoted by $U = \{u_1, \ldots, u_x\}$, the trucks by $T = \{t_1, \ldots, t_y\}$, and the packages by $P = \{p_1, \ldots, p_n\}$. Let $T_i = \{t \in T \mid I[t] \in L_i\}$ denote the trucks of city i, and $P = P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5$ denote a partition of the packages as follows:

- each package in $P_1 = \{p \in P \mid I[p], G[p] \in L^A\}$ is both initially at an airport and needs to be moved to another airport,
- each package in $P_2 = \{p \in P \mid I[p] \in L^A \cap L_i, G[p] \in L_j \setminus L^A, i \neq j\}$ is initially at an airport and needs to be moved to a non-airport location in another city,
- each package in $P_3 = \{p \in P \mid I[p] \in L_i, G[p] \in L_i\}$ needs to be moved within one city,
- each package in $P_4 = \{p \in P \mid I[p] \in L_i \setminus L^A, G[p] \in L^A \setminus L_i\}$ needs to be moved from a non-airport location in one city to the airport of some other city, and
- each package in $P_5 = \{p \in P \mid I[p] \in L_i \setminus L^A, G[p] \in L_j \setminus L^A, i \neq j\}$ needs to be moved from a non-airport location in one city to a non-airport location in another city.

A natural LOGISTICS task description in SAS$^+$ is as follows.

- Variables $V = U \cup T \cup P$ with domains

$$\forall u \in U : \mathcal{D}(u) = L^A,$$
$$\forall 1 \leq i \leq k, \forall t \in T_i : \mathcal{D}(t) = L_i,$$
$$\forall p \in P : \mathcal{D}(p) = L \cup U \cup T.$$

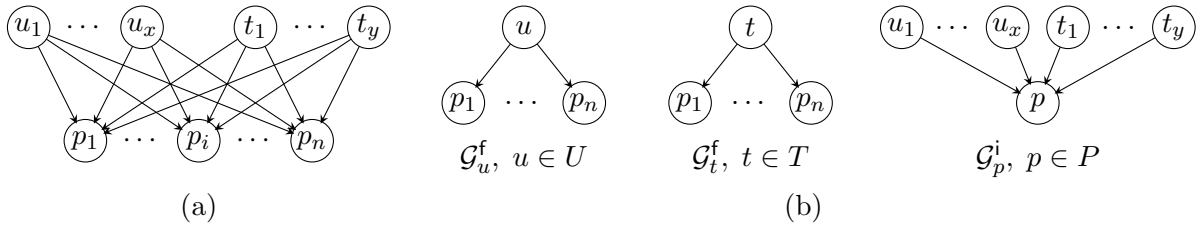


Figure 4.7: Logistics's (a) causal graph and (b) the corresponding collection of v-forks and v-iforks

- Initial state $I \in (L^A)^x \times L_1 \times \cdots \times L_k \times (L)^n$.
- Goal $G = \{p_1 = l_1, \ldots, p_n = l_n\} \in (L)^n$.
- Actions

$$A = \bigcup_{i=1}^{k} \bigcup_{l \in L_i} \bigcup_{t \in T_i} \left[\{Lt(p,t,l), Ut(p,t,l) \mid p \in P\} \cup \{Mt(t,l,l') \mid, l' \in L_i \setminus \{l\}\}\right]$$
$$\cup \bigcup_{l \in L^A} \bigcup_{u \in U} \left[\{La(p,u,l), Ua(p,u,l) \mid p \in P\} \cup \{Ma(u,l,l') \mid l' \in L^A \setminus \{l\}\}\right],$$

  where

  - load package p onto truck t in location l: $Lt(p,t,l) = \langle\{p = l, t = l\}, \{p = t\}\rangle$,
  - unload package p from truck t in location l: $Ut(p,t,l) = \langle\{p = t, t = l\}, \{p = l\}\rangle$,
  - move truck t from location l to location l': $Mt(t,l,l') = \langle\{t = l\}, \{t = l'\}\rangle$,
  - load package p onto airplane u in l: $La(p,u,l) = \langle\{p = l, u = l\}, \{p = u\}\rangle$,
  - unload package p from airplane u into l: $Ua(p,u,l) = \langle\{p = u, u = l\}, \{p = l\}\rangle$, and
  - move airplane u from location l to l': $Ma(u,l,l') = \langle\{u = l\}, \{u = l'\}\rangle$.

The (parametrized in n, x, and y) causal graph of Logistics tasks is depicted in Figure 4.7a.

Fork Decomposition Since the variables $u \in U$ and $t \in T$ have no goal value, the collection of v-forks and v-iforks is as in Figure 4.7b. The domains of the inverted-fork sinks are all abstracted as in Eq. 4.14 ("distance-from-initial-value"), p. 45, while the domains of the fork roots are abstracted as in Eq. 4.24 ("leave-one-out"), p. 69. Thus, we have

$$\boldsymbol{\Pi}_{\mathcal{F}} = \bigcup_{u \in U} \bigcup_{l \in L^A} \{\Pi^{\mathsf{f}}_{u,l}\} \cup \bigcup_{i=1}^{k} \bigcup_{t \in T_i} \bigcup_{l \in L_i} \{\Pi^{\mathsf{f}}_{t,l}\},$$
$$\boldsymbol{\Pi}_{\mathcal{I}} = \bigcup_{p \in P} \{\Pi^{\mathsf{i}}_{p,1}\} \cup \bigcup_{p \in P_2 \cup P_4 \cup P_5} \{\Pi^{\mathsf{i}}_{p,2}\} \cup \bigcup_{p \in P_5} \{\Pi^{\mathsf{i}}_{p,3}\},$$
$$\boldsymbol{\Pi}_{\mathcal{FI}} = \bigcup_{u \in U} \bigcup_{l \in L^A} \{\Pi^{\mathsf{f}}_{u,l}\} \cup \bigcup_{i=1}^{k} \bigcup_{t \in T_i} \bigcup_{l \in L_i} \{\Pi^{\mathsf{f}}_{t,l}\} \cup \bigcup_{p \in P} \{\Pi^{\mathsf{i}}_{p,1}\} \cup \bigcup_{p \in P_2 \cup P_4 \cup P_5} \{\Pi^{\mathsf{i}}_{p,2}\} \cup \bigcup_{p \in P_5} \{\Pi^{\mathsf{i}}_{p,3}\}.$$

Action	$\Pi^{\mathsf{f}}_{u,l}$	$\Pi^{\mathsf{f}}_{u,l'}$	$\Pi^{\mathsf{f}}_{u,l''}$	$\Pi^{\mathsf{f}}_{u',l}$	$\Pi^{\mathsf{f}}_{t,l}$	$\Pi^{\mathsf{f}}_{t,l'}$	$\Pi^{\mathsf{f}}_{t,l''}$	$\Pi^{\mathsf{f}}_{t',l}$	$\Pi^{\mathsf{i}}_{p,m}$	$\mathbf{\Pi}_{\mathcal{F}}$	$\mathbf{\Pi}_{\mathcal{J}}$	$\mathbf{\Pi}_{\mathcal{FJ}}$
$Mt(t,l,l')$	0	0	0	0	1	1	0	0	1	$\frac{1}{2}$	$\frac{1}{n^{\mathsf{i}}}$	$\frac{1}{2+n^{\mathsf{i}}}$
$Ma(u,l,l')$	1	1	0	0	0	0	0	0	1	$\frac{1}{2}$	$\frac{1}{n^{\mathsf{i}}}$	$\frac{1}{2+n^{\mathsf{i}}}$

(a)

					$I[p] \in L^A \cap L_i$				$I[p] \in L_i \setminus L^A$								
					$p \in P_1$	$p \in P_2$		$p \in P_3$	$p \in P_3$	$p \in P_4$		$p \in P_5$					
Action		$\Pi^{\mathsf{f}}_{u,l}$	$\Pi^{\mathsf{f}}_{t,l}$	$\Pi^{\mathsf{i}}_{p',m}$	$\Pi^{\mathsf{i}}_{p,1}$	$\Pi^{\mathsf{i}}_{p,1}$	$\Pi^{\mathsf{i}}_{p,2}$	$\Pi^{\mathsf{i}}_{p,1}$	$\Pi^{\mathsf{i}}_{p,1}$	$\Pi^{\mathsf{i}}_{p,1}$	$\Pi^{\mathsf{i}}_{p,2}$	$\Pi^{\mathsf{i}}_{p,1}$	$\Pi^{\mathsf{i}}_{p,2}$	$\Pi^{\mathsf{i}}_{p,3}$	$\mathbf{\Pi}_{\mathcal{F}}$	$\mathbf{\Pi}_{\mathcal{J}}$	$\mathbf{\Pi}_{\mathcal{FJ}}$
$Lt(p,t,l)$, $Ut(p,t,l)$	$l \in L_i$	1	1	0	1	1	0	1	1	1	0	1	0	0	$\frac{1}{n^{\mathsf{f}}}$	1	$\frac{1}{n^{\mathsf{f}}+1}$
	$l \in L_j$	1	1	0	0	0	1	0	0	0	0	0	0	1	$\frac{1}{n^{\mathsf{f}}}$	1	$\frac{1}{n^{\mathsf{f}}+1}$
$La(p,u,l)$, $Ua(p,u,l)$		1	1	0	1	1	0	1	0	0	1	0	1	0	$\frac{1}{n^{\mathsf{f}}}$	1	$\frac{1}{n^{\mathsf{f}}+1}$

(b)

Figure 4.8: Number of representatives of each original LOGISTICS action in each abstract task, as well as the partition of the action costs between these representatives; tables (a) and (b) capture the move and load/unload actions, respectively

The total number of forks is $n^{\mathsf{f}} = |\mathbf{\Pi}_{\mathcal{F}}| = |U| \cdot |L^A| + \sum_{i=1}^{k} |T_i| \cdot |L_i|$, and the total number of inverted forks is $n^{\mathsf{i}} = |\mathbf{\Pi}_{\mathcal{J}}| = |P_1| + 2 \cdot |P_2| + |P_3| + 2 \cdot |P_4| + 3 \cdot |P_5|$. For each action $a \in A$, the number of its representatives in each abstract task, as well as the cost assigned to each such representative, are given in Figure 4.8. Each row in the tables of Figure 4.8 corresponds to a certain LOGISTICS action, each column (except for the last three) represents an abstract task, and each entry captures the number of representatives an action has in the corresponding task. The last three columns show the portion of the total cost that is given to an action representative in each task, in each of the three heuristics in question.

Lower Bound Note that any optimal plan for a LOGISTICS task contains at least as many load/unload actions as move actions. Thus, the following lemma provides us with the lower bound of 1/2 for all three heuristics in question.

Lemma 2 *For any* LOGISTICS *task, $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$ account for the full cost of the load/unload actions required by any optimal plan for that task.*

Proof: For any LOGISTICS task, all the optimal plans for that task contain the same amount of load/unload actions for each package $p \in P$ as follows.

$p \in P_1$**:** 2 actions — one load onto an airplane, and one unload from that airplane,

$p \in P_2$**:** 4 actions — one load onto an airplane, one unload from that airplane, one load onto a truck, and one unload from that truck,

$p \in P_3$**:** 2 actions — one load onto a truck, and one unload from that truck,

$p \in P_4$**:** 4 actions — one load onto a truck, one unload from that truck, one load onto an airplane, and one unload from that airplane, and

$p \in P_5$**:** 6 actions — two loads onto some trucks, two unloads from these trucks, one load onto an airplane, and one unload from that airplane.

Consider the fork-decomposition $\mathbf{\Pi}_{\mathcal{F}}$. Any optimal plan for each of the abstract tasks will contain the number of load/unload actions exactly as above (the effects of these actions remain unchanged in these tasks). The cost of each representative of each load/unload action is $\frac{1}{n^{\mathsf{f}}}$, and there are n^{f} abstract tasks. Therefore, the heuristic $h^{\mathcal{F}}$ fully accounts for the cost of the required load/unload actions.

Now consider the fork-decomposition $\mathbf{\Pi}_{\mathcal{J}}$. With m being the domain-decomposition index of the abstraction, any optimal plan for the abstract task $\Pi^{\mathsf{i}}_{p,m}$ will include one load and one unload actions as follows.

$p \in P_1$**:** one load onto an airplane and one unload from that airplane,

$p \in P_2, m{=}1$**:** one load onto an airplane and one unload from that airplane,

$p \in P_2, m{=}2$**:** one load onto a truck and one unload from that truck,

$p \in P_3$**:** one load onto a truck and one unload from that truck,

$p \in P_4, m{=}1$**:** one load onto a truck and one unload from that truck,

$p \in P_4, m{=}2$**:** one load onto an airplane, and one unload from that airplane,

$p \in P_5, m{=}1$**:** one load onto a truck and one unload from that truck,

$p \in P_5, m{=}2$**:** one load onto an airplane and one unload from that airplane, and

$p \in P_5, m{=}3$**:** one load onto a truck and one unload from that truck.

The cost of each representative of load/unload actions is 1, and thus the heuristic $h^{\mathcal{J}}$ fully accounts for the cost of the required load/unload actions.

Finally, consider the fork-decomposition $\mathbf{\Pi}_{\mathcal{FJ}}$. Any optimal plan for each of the fork-structured abstract tasks will contain the same number of load/unload actions as for $\mathbf{\Pi}_{\mathcal{F}}$. The cost of each representative of load/unload actions is $\frac{1}{n^{\mathsf{f}}+1}$ and there are n^{f} such abstract tasks. In addition, each of these load/unload actions will also appear in exactly one inverted fork-structured abstract task. Therefore the heuristic $h^{\mathcal{FJ}}$ also fully accounts for the cost of the required load/unload actions. ■

Upper Bound An instance on which all three heuristics achieve exactly 1/2 consists of two trucks t_1, t_2, no airplanes, one city, and n packages such that the initial and goal locations of all the packages are all pairwise different, and both trucks are initially located at yet another location. More formally, if $L = \{l_i\}_{i=0}^{2n}$, and $T = \{t_1, t_2\}$, then the SAS$^+$ encoding for this LOGISTICS task is as follows.

- Variables $V = \{t_1, t_2, p_1, \ldots, p_n\}$ with domains

$$\begin{aligned} \forall t \in T: &\quad \mathcal{D}(t) = L, \\ \forall p \in P: &\quad \mathcal{D}(p) = L \cup T. \end{aligned}$$

- Initial state $I = \{t_1 = l_0, t_2 = l_0, p_1 = l_1, \ldots, p_n = l_n\}$.

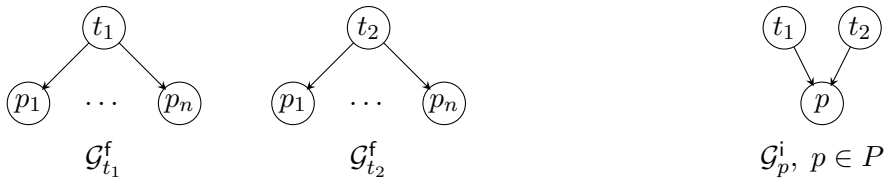


Figure 4.9: Collection of v-forks and v-iforks for the LOGISTICS task used for the proof of the upper bound of 1/2

- Goal $G = \{p_1 = l_{n+1}, \ldots, p_n = l_{2n}\}$.
- Actions $A = \{Lt(p,t,l), Ut(p,t,l) \mid l \in L, t \in T, p \in P\} \cup \{Mt(t,l,l') \mid t \in T, \{l,l'\} \subseteq L\}$.

The collection of v-forks and v-iforks for this task is depicted in Figure 4.9. The domains of the inverted-fork sinks are all abstracted as in Eq. 4.14 ("distance-from-initial-value"), while the domains of the fork roots are abstracted as in Eq. 4.24 ("leave-one-out"), and therefore we have

$$\begin{aligned}
\mathbf{\Pi}_{\mathcal{F}} &= \{\Pi^{\mathsf{f}}_{t_1,l}\Pi^{\mathsf{f}}_{t_2,l} \mid l \in L\},\\
\mathbf{\Pi}_{\mathcal{J}} &= \{\Pi^{\mathsf{i}}_{p,1} \mid p \in P\},\\
\mathbf{\Pi}_{\mathcal{FJ}} &= \{\Pi^{\mathsf{f}}_{t_1,l}\Pi^{\mathsf{f}}_{t_2,l} \mid l \in L\} \cup \{\Pi^{\mathsf{i}}_{p,1} \mid p \in P\}.
\end{aligned}$$

The total number of forks is thus $n^{\mathsf{f}} = 4n+2$ and the total number of inverted forks is $n^{\mathsf{i}} = n$. The partition of the action costs for LOGISTICS tasks is described in Figure 4.8. Here we have $P = P_3$ and thus the action cost partition is as follows.

Action	$\Pi^{\mathsf{f}}_{t,l}$	$\Pi^{\mathsf{f}}_{t,l'}$	$\Pi^{\mathsf{f}}_{t,l''}$	$\Pi^{\mathsf{f}}_{t',l^*}$	$\Pi^{\mathsf{i}}_{p,1}$	$\Pi^{\mathsf{i}}_{p',1}$	$\mathbf{\Pi}_{\mathcal{F}}$	$\mathbf{\Pi}_{\mathcal{J}}$	$\mathbf{\Pi}_{\mathcal{FJ}}$
$Mt(t,l,l')$	1	1	0	0	1	0	$\frac{1}{2}$	$\frac{1}{n}$	$\frac{1}{n+2}$
$Lt(p,t,l)$	1	1	1	1	1	0	$\frac{1}{4n+2}$	1	$\frac{1}{4n+3}$
$Ut(p,t,l)$	1	1	1	1	1	0	$\frac{1}{4n+2}$	1	$\frac{1}{4n+3}$

Given that, the optimal plans for the abstract task are

h	task	optimal plan	cost	#	$h(I)$
$h^{\mathcal{F}}$	$\Pi^{\mathsf{f}}_{t_1,l}$	$\langle Lt(p_1,t_2,l_1) \cdot \ldots \cdot Lt(p_n,t_2,l_n) \cdot Ut(p_1,t_2,l_{n+1}) \cdot \ldots \cdot Ut(p_n,t_2,l_{2n})\rangle$	$\frac{2n}{4n+2}$	$2n+1$	$2n$
	$\Pi^{\mathsf{f}}_{t_2,l}$	$\langle Lt(p_1,t_1,l_1) \cdot \ldots \cdot Lt(p_n,t_1,l_n) \cdot Ut(p_1,t_1,l_{n+1}) \cdot \ldots \cdot Ut(p_n,t_1,l_{2n})\rangle$	$\frac{2n}{4n+2}$	$2n+1$	
$h^{\mathcal{J}}$	$\Pi^{\mathsf{i}}_{p_i,1}$	$\langle Mt(t_1,l_0,l_i) \cdot Lt(p_i,t_1,l_i) \cdot Mt(t_1,l_i,l_{n+i}) \cdot Ut(p_i,t_1,l_{n+i})\rangle$	$\frac{2}{n}+2$	n	$2n+2$
$h^{\mathcal{FJ}}$	$\Pi^{\mathsf{f}}_{t_1,l}$	$\langle Lt(p_1,t_2,l_1) \cdot \ldots \cdot Lt(p_n,t_2,l_n) \cdot Ut(p_1,t_2,l_{n+1}) \cdot \ldots \cdot Ut(p_n,t_2,l_{2n})\rangle$	$\frac{2n}{4n+3}$	$2n+1$	$2n+\frac{2n}{n+2}$
	$\Pi^{\mathsf{f}}_{t_2,l}$	$\langle Lt(p_1,t_1,l_1) \cdot \ldots \cdot Lt(p_n,t_1,l_n) \cdot Ut(p_1,t_1,l_{n+1}) \cdot \ldots \cdot Ut(p_n,t_1,l_{2n})\rangle$	$\frac{2n}{4n+3}$	$2n+1$	
	$\Pi^{\mathsf{i}}_{p_i,1}$	$\langle Mt(t_1,l_0,l_i) \cdot Lt(p_i,t_1,l_i) \cdot Mt(t_1,l_i,l_{n+i}) \cdot Ut(p_i,t_1,l_{n+i})\rangle$	$\frac{2}{n+2}+\frac{2}{4n+3}$	n	

while an optimal plan for the original task, e.g., $\langle Mt(t_1,l_0,l_1) \cdot Lt(p_1,t_1,l_1) \cdot Mt(t_1,l_1,l_2) \cdot Lt(p_2,t_1,l_2) \cdot Mt(t_1,l_2,l_3) \cdot \ldots \cdot Lt(p_n,t_1,l_n) \cdot Mt(t_1,l_n,l_{n+1}) \cdot Ut(p_1,t_1,l_{n+1}) \cdot Mt(t_1,l_{n+1},l_{n+2}) \cdot Ut(p_2,t_1,l_{n+2}) \cdot Mt(t_1,l_{n+2},l_{n+3}) \cdot \ldots Ut(p_n,t_1,l_{2n})\rangle$, has the cost of $4n$, providing us with the upper bound of 1/2 for all three heuristics. Putting our lower and upper bounds together, the asymptotic ratio of all three heuristics in question is 1/2.

Blocksworld

Each BLOCKSWORLD task consists of a table table, a crane c, and $n+1$ blocks $B = \{b_1, \ldots, b_{n+1}\}$. Each block can be either on the table, or on top of some other block, or held by the crane. The crane can pick up a block if it currently holds nothing, and that block has no other block on top of it. The crane can drop the held block on the table or on top of some other block.

Consider now a BLOCKSWORLD task as follows. The blocks initially form a tower $b_1, \ldots, b_n, b_{n+1}$ with b_{n+1} being on the table, and the goal is to move them to form a tower $b_1, \ldots, b_{n-1}, b_{n+1}, b_n$ with b_n being on the table. That is, the goal is to swap the lowest two blocks of the tower. A natural description of this task in SAS$^+$ is as follows.

- Variables $V = \{b, clear_b \mid b \in B\} \cup \{c\}$ with domains

$$\begin{aligned} \mathcal{D}(c) &= \{\mathsf{empty}\} \cup B, \\ \forall b \in B: \ \mathcal{D}(b) &= \{\mathsf{table}, \mathsf{c}\} \cup B \setminus \{b\}, \\ \mathcal{D}(clear_b) &= \{\mathsf{yes}, \mathsf{no}\}. \end{aligned}$$

- Initial state

$$\begin{aligned} I = \ & \{c = \mathsf{empty}, b_{n+1} = \mathsf{table}, clear_{b_1} = \mathsf{yes}\} \bigcup \\ & \{b_i = b_{i+1} \mid 1 \leq i \leq n\} \bigcup \\ & \{clear_b = \mathsf{no} \mid b \in B \setminus \{b_1\}\}. \end{aligned}$$

- Goal $G = \{b_n = \mathsf{table}, b_{n+1} = b_n, b_{n-1} = b_{n+1}\} \cup \{b_i = b_{i+1} \mid 1 \leq i \leq n-2\}$.
- Actions $A = \{P_T(b), D_T(b) \mid b \in B\} \cup \{P(b, b'), D(b, b') \mid \{b, b'\} \subseteq B\}$ where
  - pick block b from the table: $P_T(b) = \langle \{c{=}\mathsf{empty}, b{=}\mathsf{table}, clear_b{=}\mathsf{yes}\}, \{c{=}\mathsf{b}, b{=}\mathsf{c}\} \rangle$,
  - pick block b from block b': $P(b, b') = \langle \{c{=}\mathsf{empty}, b{=}\mathsf{b}', clear_b{=}\mathsf{yes}, clear_{b'}{=}\mathsf{no}\}, \{c{=}\mathsf{b}, b{=}\mathsf{c}, clear_{b'}{=}\mathsf{yes}\} \rangle$,
  - drop block b on the table: $D_T(b) = \langle \{c{=}\mathsf{b}, b{=}\mathsf{c}\}, \{c{=}\mathsf{empty}, b{=}\mathsf{table}\} \rangle$, and
  - drop block b on block b': $D(b, b') = \langle \{c{=}\mathsf{b}, b{=}\mathsf{c}, clear_{b'}{=}\mathsf{yes}\}, \{c{=}\mathsf{empty}, b{=}\mathsf{b}', clear_{b'}{=}\mathsf{no}\} \rangle$.

A schematic version of the causal graph of this task is depicted in Figure 4.10a. Since only the variables b_{n-1}, b_n, b_{n+1} have goal values that are different from their values in the initial state, the collection of v-forks and v-iforks is as in Figure 4.10b. After the ("leave-one-out," Eq. 4.24) domain abstraction of the variable c, c-fork $\mathcal{G}^{\mathsf{f}}_c$ breaks down into $n+2$ abstract tasks. The sinks of v-iforks $\mathcal{G}^{\mathsf{i}}_{b_{n-1}}$, $\mathcal{G}^{\mathsf{i}}_{b_n}$, and $\mathcal{G}^{\mathsf{i}}_{b_{n+1}}$ also go through the process of domain decomposition ("distance-from-initial-value," Eq. 4.14). However, due to the structure of the domain transition graphs of the block variables, domain decomposition here results in only a single abstract task for each of the v-iforks. Thus we have

$$\begin{aligned} \boldsymbol{\Pi}_{\mathcal{F}} &= \{\Pi^{\mathsf{f}}_{c,\mathsf{empty}}\} \cup \{\Pi^{\mathsf{f}}_{c,b} \mid b \in B\} \cup \{\Pi^{\mathsf{f}}_{clear_b} \mid b \in B\}, \\ \boldsymbol{\Pi}_{\mathcal{I}} &= \{\Pi^{\mathsf{i}}_{b_{n-1},1}, \Pi^{\mathsf{i}}_{b_n,1}, \Pi^{\mathsf{i}}_{b_{n+1},1}\}, \\ \boldsymbol{\Pi}_{\mathcal{FI}} &= \{\Pi^{\mathsf{f}}_{c,\mathsf{empty}}\} \cup \{\Pi^{\mathsf{f}}_{c,b} \mid b \in B\} \cup \{\Pi^{\mathsf{f}}_{clear_b} \mid b \in B\} \cup \Pi^{\mathsf{i}}_{b_{n-1},1}, \Pi^{\mathsf{i}}_{b_n,1}, \Pi^{\mathsf{i}}_{b_{n+1},1}\}. \end{aligned}$$

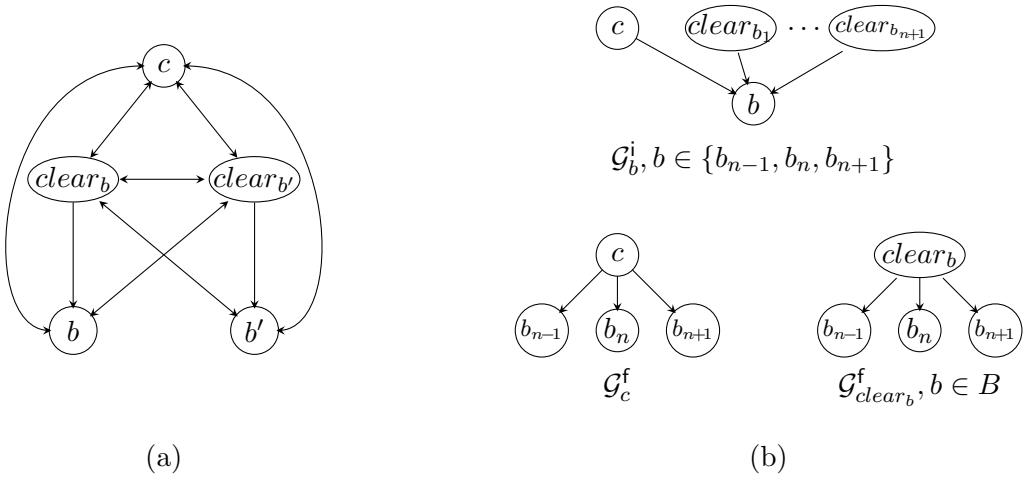


Figure 4.10: (a) Causal graph and (b) the corresponding collection of v-forks and v-iforks for the BLOCKSWORLD task used in the proof

It is technically straightforward to verify that, for each abstract task in $\mathbf{\Pi}_{\mathcal{F}}$, $\mathbf{\Pi}_{\mathcal{J}}$, and $\mathbf{\Pi}_{\mathcal{FJ}}$, there exists a plan that (i) involves only the representatives of the actions

$$\{P(b_{n-1}, b_n), D_T(b_{n-1}), P(b_n, b_{n+1}), D_T(b_n), P_T(b_{n+1}), D(b_{n+1}, b_n), P_T(b_{n-1}), D(b_{n-1}, b_{n+1})\}, \tag{4.15}$$

and (ii) involves each representative of each original action at most once. Even if together these plans account for the total cost of all eight actions in Eq. 4.15, the total cost of all these plans (and thus the estimates of all the three heuristics) is upper-bounded by 8, while an optimal plan for the original task, e.g., $\langle P(b_1, b_2) \cdot D_T(b_1) \cdot P(b_2, b_3) \cdot D_T(b_2) \cdot \ldots \cdot P(b_n, b_{n+1}) \cdot D_T(b_n) \cdot P_T(b_{n+1}) \cdot D(b_{n+1}, b_n) \cdot P_T(b_{n-1}) \cdot D(b_{n-1}, b_{n+1}) \cdot P_T(b_{n-2}) \cdot D(b_{n-2}, b_{n-1}) \cdot \ldots \cdot P_T(b_1) \cdot D(b_1, b_2)\rangle$, has a cost of $4n$. Hence, the asymptotic performance ratio of all three heuristics on the BLOCKSWORLD domain is 0.

Miconic

Each MICONIC task consists of one elevator e, a set of floors F, and the passengers P. The elevator can move between $|F|$ floors and on each floor it can load and/or unload passengers. A natural SAS$^+$ description of a MICONIC task is as follows.

- Variables $V = \{e\} \cup P$ with domains

$$\begin{aligned} \mathcal{D}(e) &= F, \\ \forall p \in P: \quad \mathcal{D}(p) &= F \cup \{\mathsf{e}\}. \end{aligned}$$

- Initial state $I = \{e = f_e\} \cup \{p = f_p \mid p \in P\} \in (F)^{|P|+1}$.
- Goal $G = \{p = f'_p \mid p \in P\} \in (F)^{|P|}$.
- Actions $A = \{In(p, f), Out(p, f) \mid f \in F, p \in P\} \cup \{Move(f, f') \mid \{f, f'\} \subseteq F\}$, where
  - load passenger p into e on floor f: $In(p, f) = \langle \{e = f, p = f\}, \{p = \mathsf{e}\} \rangle$,

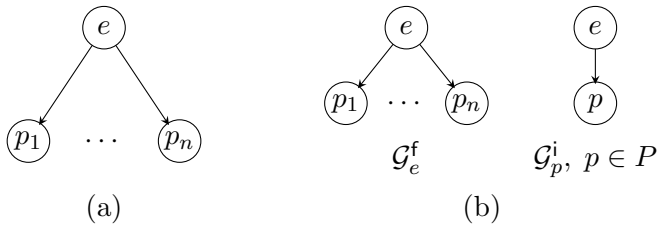


Figure 4.11: MICONIC's (a) causal graph and (b) the corresponding collection of v-forks and v-iforks

Action	$\Pi^{\mathsf{f}}_{e,f}$	$\Pi^{\mathsf{f}}_{e,f'}$	$\Pi^{\mathsf{f}}_{e,f''}$	$\Pi^{\mathsf{i}}_{p,1}$	$\Pi^{\mathsf{i}}_{p',1}$	$\mathbf{\Pi}_{\mathcal{F}}$	$\mathbf{\Pi}_{\mathcal{J}}$	$\mathbf{\Pi}_{\mathcal{FJ}}$
$Move(f,f')$	1	1	0	1	1	$\frac{1}{2}$	$\frac{1}{n^{\mathsf{i}}}$	$\frac{1}{2+n^{\mathsf{i}}}$
$In(p,f)$	1	1	1	1	0	$\frac{1}{n^{\mathsf{f}}}$	1	$\frac{1}{n^{\mathsf{f}}+1}$
$In(p',f)$	1	1	1	0	1	$\frac{1}{n^{\mathsf{f}}}$	1	$\frac{1}{n^{\mathsf{f}}+1}$
$Out(p,f)$	1	1	1	1	0	$\frac{1}{n^{\mathsf{f}}}$	1	$\frac{1}{n^{\mathsf{f}}+1}$
$Out(p',f)$	1	1	1	0	1	$\frac{1}{n^{\mathsf{f}}}$	1	$\frac{1}{n^{\mathsf{f}}+1}$

Table 4.4: Number of representatives for each original MICONIC action in each abstract task, as well as the partition of the action costs among these representatives

- unload passenger p from e to floor f: $Out(p,f) = \langle\{e = f, p = \mathsf{e}\}, \{p = f\}\rangle$, and
- move elevator from floor f to floor f': $Move(f,f') = \langle\{e = f\}, \{e = f'\}\rangle$.

The (parametrized in n) causal graph of MICONIC tasks is depicted in Figure 4.11a, and Figure 4.11b depicts the corresponding collection of v-forks and v-iforks. The domains of the inverted-fork sinks are all abstracted as in Eq. 4.14 ("distance-from-initial-value"), and the domains of the fork roots are abstracted as in Eq. 4.24 ("leave-one-out"). Thus, we have

$$\begin{aligned}
\mathbf{\Pi}_{\mathcal{F}} &= \{\Pi^{\mathsf{f}}_{e,f} \mid f \in F\},\\
\mathbf{\Pi}_{\mathcal{J}} &= \{\Pi^{\mathsf{i}}_{p,1} \mid p \in P\},\\
\mathbf{\Pi}_{\mathcal{FJ}} &= \{\Pi^{\mathsf{f}}_{e,f} \mid f \in F\} \cup \{\Pi^{\mathsf{i}}_{p,1} \mid p \in P\}.
\end{aligned}$$

The total number of the fork-structured abstract tasks is thus $n^{\mathsf{f}} = |\mathbf{\Pi}_{\mathcal{F}}| = |F|$ and the total number of the inverted fork structured abstract tasks is $n^{\mathsf{i}} = |\mathbf{\Pi}_{\mathcal{J}}| = |P|$. For each action $a \in A$, the number of its representatives in each abstract task, as well as the cost assigned to each such representative, are given in Table 4.4.

Lower Bounds First, as MICONIC is a special case of the LOGISTICS domain, Lemma 2 applies here analogously, with each package in P_3 corresponding to a passenger. Thus, for each $p \in P$, all three heuristics account for the full cost of the load/unload actions required by any optimal plan for that task.

Let us now focus on the abstract tasks $\mathbf{\Pi}_{\mathcal{F}} = \{\Pi^{\mathsf{f}}_{e,f} \mid f \in F\}$. Recall that the task $\Pi^{\mathsf{f}}_{e,f}$ is induced by an e-fork and, in terms of domain decomposition, distinguishes between being at floor f and being somewhere else. Without loss of generality, the set of floors F can be restricted to the initial and the goal values of the variables, and this because no optimal plan will move the elevator

to or from a floor f that is neither an initial nor a goal location of a passenger or the elevator. Let $F_I = \{I[p] \mid p \in P\}$ and $F_G = \{G[p] \mid p \in P\}$. The costs of the optimal plans for each abstract task $\Pi^{\mathsf{f}}_{e,f}$ are as follows.

$f \in F_I \cap F_G$: Let $p, p' \in P$ be a pair of passengers with initial and goal locations in f, respectively; that is, $I[p] = G[p'] = f$. If $f = I[e]$, then any plan for $\Pi^{\mathsf{f}}_{e,f}$ has to move the elevator from f in order to load passenger p', and then move the elevator back to f in order to unload passenger p'. Therefore the cost of any plan for $\Pi^{\mathsf{f}}_{e,f}$ is at least $\frac{2|P|}{|F|} + 1$, where (see the last three columns of Table 4.4) the first component of the summation comes from summing the costs of the representatives of the load/unload actions for all the passengers, and the second component is the sum of the costs of representatives of the two respective move actions. Similarly, if $f \neq I[e]$, then any plan for $\Pi^{\mathsf{f}}_{e,f}$ has to move the elevator to f in order to load passenger p, and then move the elevator from f in order to unload p. Therefore, here as well, the cost of any plan for $\Pi^{\mathsf{f}}_{e,f}$ is at least $\frac{2|P|}{|F|} + 1$.

$f \in F_I \setminus F_G$: Let $p \in P$ be a passenger initially at f, that is, $I[p] = f$. If $f = I[e]$, then any plan for $\Pi^{\mathsf{f}}_{e,f}$ has to move the elevator from f in order to unload p, and thus the cost of any plan for $\Pi^{\mathsf{f}}_{e,f}$ is at least $\frac{2|P|}{|F|} + \frac{1}{2}$. Otherwise, if $f \neq I[e]$, then any plan for $\Pi^{\mathsf{f}}_{e,f}$ has to move the elevator to f in order to load p, and then move the elevator from f in order to unload p. Hence, in this case, the cost of any plan for $\Pi^{\mathsf{f}}_{e,f}$ is at least $\frac{2|P|}{|F|} + 1$.

$f \in F_G \setminus F_I$: Let $p \in P$ be a passenger who must arrive at floor f, that is, $G[p] = f$. If $f = I[e]$, then any plan for $\Pi^{\mathsf{f}}_{e,f}$ has to move the elevator from f in order to load p, and then move the elevator back to f in order to unload p. Hence, here as well, the cost of any plan for $\Pi^{\mathsf{f}}_{e,f}$ is at least $\frac{2|P|}{|F|} + 1$. Otherwise, if $f \neq I[e]$, then any plan for $\Pi^{\mathsf{f}}_{e,f}$ has to move the elevator to f in order to unload p, and thus the cost of any plan for $\Pi^{\mathsf{f}}_{e,f}$ is at least $\frac{2|P|}{|F|} + \frac{1}{2}$.

$f \notin F_G \cup F_I$: If $f = I[e]$, then any plan for $\Pi^{\mathsf{f}}_{e,f}$ has to include a move from f in order to load/unload the passengers, and thus the cost of any plan for $\Pi^{\mathsf{f}}_{e,f}$ is at least $\frac{2|P|}{|F|} + \frac{1}{2}$. Otherwise, if $f \neq I[e]$, the elevator is initially "in the set of all other locations," and thus the cost of any plan for $\Pi^{\mathsf{f}}_{e,f}$ is at least $\frac{2|P|}{|F|}$.

Putting this case-by-case analysis together, we have

$$h^{\mathcal{F}}(I) \geq \begin{cases} 2|P| + |F_I \cap F_G| + |F_I \setminus F_G| + \frac{|F_G \setminus F_I|}{2}, & I[e] \in F_I \cap F_G \\ 2|P| + |F_I \cap F_G| + |F_I \setminus F_G| - 1 + \frac{1}{2} + \frac{|F_G \setminus F_I|}{2}, & I[e] \in F_I \setminus F_G \\ 2|P| + |F_I \cap F_G| + |F_I \setminus F_G| + 1 + \frac{|F_G \setminus F_I| - 1}{2}, & I[e] \in F_G \setminus F_I \\ 2|P| + |F_I \cap F_G| + |F_I \setminus F_G| + \frac{|F_G \setminus F_I| - 1}{2} + \frac{1}{2}, & I[e] \notin F_G \cup F_I \end{cases}.$$

Note that the value in the second case is the lowest. This gives us a lower bound on the $h^{\mathcal{F}}$ estimate as in Eq. 4.16.

$$h^{\mathcal{F}}(I) \geq 2|P| + |F_I \setminus F_G| + \frac{|F_G \setminus F_I|}{2} + |F_I \cap F_G| - \frac{1}{2}. \tag{4.16}$$

Now, let us provide an upper bound on the length (= cost) of the optimal plan for a Miconic task. First, let $P' \subseteq P$ denote the set of passengers with both initial and goal locations in $F_I \cap F_G$.

Let $m(P', F_I \cap F_G)$ denote the length of the optimal traversal of the floors $F_I \cap F_G$ such that, for each passenger $p \in P'$, a visit of $I[p]$ comes before some visit of $G[p]$. Given that, on a case-by-case basis, a (not necessarily optimal) plan for the MICONIC task at hand is as follows.

$I[e] \in F_I \cap F_G$: Collect all the passengers at $I[e]$ if any, then traverse all the floors in $F_I \setminus F_G$ and collect passengers from these floors, then move the elevator to the first floor f on the optimal path π traversing the floors $F_I \cap F_G$, drop off the passengers whose destination is f, collect the new passengers if any, keep moving along π while collecting and dropping off passengers at their initial and target floors, and then traverse $F_G \setminus F_I$, dropping off the remaining passengers at their destinations. The cost of such a plan (and thus of the optimal plan) is upper-bounded as in Eq. 4.17 below.

$$h^*(I) \leq 2|P| + |F_I \setminus F_G| + m(P', F_I \cap F_G) + |F_G \setminus F_I|. \tag{4.17}$$

$I[e] \in F_I \setminus F_G$: Collect all the passengers at $I[e]$ if any, then traverse all the floors in $F_I \setminus F_G$ and collect passengers from these floors while making sure that this traversal ends up at the first floor f of the optimal path π traversing the floors $F_I \cap F_G$, then follow π while collecting and dropping passengers off at their initial and target floors, and then traverse $F_G \setminus F_I$, dropping the remaining passengers off at their destinations. As in the first case, the cost of such a plan is upper-bounded as in Eq. 4.17.

$I[e] \notin F_I$: Traverse the floors $F_I \setminus F_G$ and collect all the passengers from these floors, then move along the optimal path π traversing the floors $F_I \cap F_G$ while collecting and dropping off passengers at their initial and target floors, and then traverse the floors $F_G \setminus F_I$, dropping the remaining passengers off at their destinations. Here as well, the cost of such a plan is upper-bounded by the expression in Eq. 4.17.

Lemma 3 *For any* MICONIC *task with passengers* P*, we have* $\frac{h^{\mathcal{F}}(I)}{h^*(I)} \geq \frac{5|P|-1}{6|P|}$.

Proof: Recall that $P' \subseteq P$ is the set of all passengers with both initial and goal locations in $F_I \cap F_G$. First we give two upper bounds on the length of the optimal traversal of the floors $F_I \cap F_G$ such that, for each passenger $p \in P'$, a visit of $I[p]$ comes before some visit of $G[p]$. From Theorem 5.3.3 of Helmert (2008) we have

$$m(P', F_I \cap F_G) = |F_I \cap F_G| + m^*(\mathcal{G}'), \tag{4.18}$$

where $m^*(\mathcal{G}')$ is the size of the minimum feedback vertex set of the directed graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, with $\mathcal{V}' = F_I \cap F_G$ and $\mathcal{E}'$ containing an arc from f to f' if and only if a passenger $p \in P'$ is initially at floor f and should arrive at floor f'.

Note that $m^*(\mathcal{G}')$ is trivially bounded by the number of graph nodes $\mathcal{V}'$. In addition, observe that, for any order of the nodes $\mathcal{V}'$, the arcs $\mathcal{E}'$ can be partitioned into "forward" and "backward" arcs, and one of these subsets must contain no more than $\frac{|\mathcal{E}'|}{2}$ arcs. Removing from $\mathcal{G}'$ all the nodes that are origins of the arcs in that smaller subset of $\mathcal{E}'$ results in a directed acyclic graph. Hence, the set of removed nodes is a (not necessarily minimum) feedback vertex set of $\mathcal{G}'$, and the size of this set is no larger than $\frac{|\mathcal{E}'|}{2}$. Putting these two bounds on $m^*(\mathcal{G}')$ together with Eq. 4.18 we obtain

$$m(P', F_I \cap F_G) \leq \min \left\{ 2|F_I \cap F_G|, \;\; |F_I \cap F_G| + \frac{|P'|}{2} \right\}. \tag{4.19}$$

From the disjointness of $F_G \setminus F_I$ and $F_I \cap F_G$, and the fact that the goal of all the passengers in P' is in F_I, we have $|F_G \setminus F_I| \leq |P| - |P'|$. From Eqs. 4.16 and 4.17 we have

$$\frac{h^{\mathcal{F}}}{h^*} \geq \frac{2|P| + |F_I \setminus F_G| + \frac{|F_G \setminus F_I|}{2} + |F_I \cap F_G| - \frac{1}{2}}{2|P| + |F_I \setminus F_G| + |F_G \setminus F_I| + m(P', F_I \cap F_G)}. \tag{4.20}$$

As we are interested in a lower bound on the ratio $\frac{h^{\mathcal{F}}}{h^*}$, the right-hand side of the inequality should be minimized, and thus we can safely set $|F_I \setminus F_G| = 0$ and $|F_G \setminus F_I| = |P| - |P'|$, obtaining

$$\frac{h^{\mathcal{F}}}{h^*} \geq \frac{2|P| + \frac{|P|-|P'|}{2} + |F_I \cap F_G| - \frac{1}{2}}{2|P| + |P| - |P'| + m(P', F_I \cap F_G)} = \frac{5|P| - |P'| + 2|F_I \cap F_G| - 1}{6|P| - 2|P'| + 2m(P', F_I \cap F_G)}. \tag{4.21}$$

Let us examine the right-most expression in Eq. 4.21 with respect to the two upper bounds on $m(P', F_I \cap F_G)$ as in Eq. 4.19.

- If the minimum is obtained on $2|F_I \cap F_G|$, then $m(P', F_I \cap F_G) \leq 2|F_I \cap F_G| \leq |F_I \cap F_G| + \frac{|P'|}{2}$, where the last inequality can be reformulated as

  $$2|F_I \cap F_G| - |P'| \leq 0.$$

  This allows us to provide a lower bound on the right-most expression in Eq. 4.21, and thus on $\frac{h^{\mathcal{F}}}{h^*}$ as

  $$\frac{h^{\mathcal{F}}}{h^*} \geq \frac{5|P| - |P'| + 2|F_I \cap F_G| - 1}{6|P| - 2|P'| + 2m(P', F_I \cap F_G)} \geq \frac{5|P| + (2|F_I \cap F_G| - |P'|) - 1}{6|P| + 2(2|F_I \cap F_G| - |P'|)} \geq \frac{5|P| - 1}{6|P|}. \tag{4.22}$$

- If the minimum is obtained on $|F_I \cap F_G| + \frac{|P'|}{2}$, then $m(P', F_I \cap F_G) \leq |F_I \cap F_G| + \frac{|P'|}{2} < 2|F_I \cap F_G|$, where the last inequality can be reformulated as

  $$2|F_I \cap F_G| - |P'| > 0.$$

  This again allows us to provide a lower bound on $\frac{h^{\mathcal{F}}}{h^*}$ via Eq. 4.21 as

  $$\frac{h^{\mathcal{F}}}{h^*} \geq \frac{5|P| - |P'| + 2|F_I \cap F_G| - 1}{6|P| - 2|P'| + 2m(P', F_I \cap F_G)} \geq \frac{5|P| + (2|F_I \cap F_G| - |P'|) - 1}{6|P| + (2|F_I \cap F_G| - |P'|)} \geq \frac{5|P| - 1}{6|P|}. \tag{4.23}$$

Note that both lower bounds on $\frac{h^{\mathcal{F}}}{h^*}$ in Eq. 4.22 and Eq. 4.23 are as required by the claim of the lemma. ∎

Upper Bounds A MICONIC task on which the heuristic $h^{\mathcal{F}}$ achieves the performance ratio of exactly 5/6 consists of an elevator e, floors $F = \{f_i\}_{i=0}^n$, passengers $P = \{p_i\}_{i=1}^n$, all the passengers and the elevator being initially at f_0, and the target floors of the passengers all being pairwise disjoint. The SAS$^+$ encoding for the MICONIC task is as follows.

- Variables $V = \{e\} \cup P$ with the domains $\mathcal{D}(e) = F$ and $\forall p \in P : \mathcal{D}(p) = F \cup \{\mathsf{e}\}$.
- Initial state $I = \{e = f_0, p_1 = f_0, \ldots, p_n = f_0\}$.

- Goal $G = \{p_1 = f_1, \ldots, p_n = f_n\}$.
- Actions $A = \{In(p, f), Out(p, f) \mid f \in F, p \in P\} \cup \{Move(f, f') \mid \{f, f'\} \subseteq F\}$.

The causal graph of this task and the corresponding collection of v-forks (consisting of only one e-fork) are depicted in Figure 4.11. The domain of e is abstracted as in Eq. 4.24 ("leave-one-out"), providing us with

$$\mathbf{\Pi}_{\mathcal{F}} = \{\Pi^{\mathsf{f}}_{e,f_0}, \Pi^{\mathsf{f}}_{e,f_1}, \ldots, \Pi^{\mathsf{f}}_{e,f_n}\}.$$

The costs of the action representatives in these abstract tasks are given in Table 4.4 with $n^{\mathsf{f}} = n+1$. The optimal plans for the abstract tasks in $\mathbf{\Pi}_{\mathcal{F}}$ are

task	optimal plan	cost	#	$h^{\mathcal{F}}(I)$
Π^{f}_{e,f_0}	$\langle In(p_1,f_0)\cdot\ldots\cdot In(p_n,f_0)\cdot Move(f_0,f_1)\cdot Out(p_1,f_1)\cdot\ldots\cdot Out(p_n,f_n)\rangle$	$\frac{1}{2}+\frac{2n}{n+1}$	$n+1$	$\frac{5n+1}{2}$
Π^{f}_{e,f_1}	$\langle In(p_1,f_0)\cdot\ldots\cdot In(p_n,f_0)\cdot Out(p_2,f_2)\cdot\ldots\cdot Out(p_n,f_n)\cdot Move(f_0,f_1)\cdot Out(p_1,f_1)\rangle$	$\frac{1}{2}+\frac{2n}{n+1}$		
Π^{f}_{e,f_n}	$\langle In(p_1,f_0)\cdot\ldots\cdot In(p_n,f_0)\cdot Out(p_1,f_1)\cdot\ldots\cdot Out(p_{n-1},f_{n-1})\cdot Move(f_0,f_n)\cdot Out(p_n,f_n)\rangle$	$\frac{1}{2}+\frac{2n}{n+1}$		

while an optimal plan for the original task, $\langle In(p_1,f_0)\cdot\ldots\cdot In(p_n,f_0)\cdot Move(f_0,f_1)\cdot Out(p_1,f_1)\cdot Move(f_1,f_2)\cdot Out(p_2,f_2)\cdot Move(f_2,f_3)\cdot\ldots\cdot Out(p_n,f_n)\rangle$, has a cost of $3n$, providing us with the upper bound of $5/6$ for the $h^{\mathcal{F}}$ heuristic in MICONIC. Putting this upper bound together with the previously obtained lower bound of $5/6$, we conclude that the asymptotic performance ratio of $h^{\mathcal{F}}$ in MICONIC is $5/6$.

A MICONIC task on which the heuristics $h^{\mathcal{J}}$ and $h^{\mathcal{FJ}}$ achieve exactly $1/2$ consists of an elevator e, floors $F = \{f_i\}_{i=0}^{2n}$, passengers $P = \{p_i\}_{i=1}^{n}$, and the initial and target floors for all the passengers and the elevator being pairwise disjoint. The task description in SAS$^+$ is as follows.

- Variables $V = \{e\} \cup P$ with the domains $\mathcal{D}(e) = F$ and $\forall p \in P : \mathcal{D}(p) = F \cup \{\mathsf{e}\}$.
- Initial state $I = \{e = f_0, p_1 = f_1, \ldots, p_n = f_n\}$.
- Goal $G = \{p_1 = f_{n+1}, \ldots, p_n = f_{2n}\}$.
- Actions $A = \{In(p, f), Out(p, f) \mid f \in F, p \in P\} \cup \{Move(f, f') \mid \{f, f'\} \subseteq F\}$.

The causal graph of this task and the corresponding collection of v-forks and v-iforks are depicted in Figure 4.11. The domains of the inverted-fork sinks are all abstracted as in Eq. 4.14 ("distance-from-initial-value"), and the domains of the fork roots are all abstracted as in Eq. 4.24 ("leave-one-out"). This provides us with

$$\mathbf{\Pi}_{\mathcal{J}} = \{\Pi^{\mathsf{i}}_{p_1,1}, \ldots, \Pi^{\mathsf{i}}_{p_n,1}\},$$
$$\mathbf{\Pi}_{\mathcal{FJ}} = \{\Pi^{\mathsf{f}}_{e,f_0}, \Pi^{\mathsf{f}}_{e,f_1}, \ldots, \Pi^{\mathsf{f}}_{e,f_n}, \Pi^{\mathsf{f}}_{e,f_{n+1}}, \ldots, \Pi^{\mathsf{f}}_{e,f_{2n}}, \Pi^{\mathsf{i}}_{p_1,1}, \ldots, \Pi^{\mathsf{i}}_{p_n,1}\}.$$

The costs of the action representatives in these abstract tasks are given in Table 4.4 with $n^{\mathsf{f}} = 2n+1$ and $n^{\mathsf{i}} = n$. The optimal plans for the abstract tasks in $\mathbf{\Pi}_{\mathcal{J}}$ and $\mathbf{\Pi}_{\mathcal{FJ}}$ are

h	task	optimal plan	cost	#	$h(I)$
$h^{\mathcal{J}}$	$\Pi^{\mathsf{i}}_{p_i,1}$	$\langle Move(f_0,f_i)\cdot In(p_i,f_i)\cdot Move(f_i,f_{n+i})\cdot Out(p_i,f_{n+i})\rangle$	$\frac{2}{n}+2$	n	$2n+2$
$h^{\mathcal{FJ}}$	Π^{f}_{e,f_0}	$\langle Move(f_0,f_1)\cdot In(p_1,f_1)\cdot\ldots\cdot In(p_n,f_n)\cdot Out(p_1,f_{n+1})\cdot\ldots\cdot Out(p_n,f_{2n})\rangle$	$\frac{1}{n+2}+\frac{2n}{2n+2}$	1	$2n+\frac{5n+1}{n+2}$
	Π^{f}_{e,f_1}	$\langle Move(f_0,f_1)\cdot In(p_1,f_1)\cdot Move(f_1,f_2)\cdot In(p_2,f_2)\cdot\ldots\cdot In(p_n,f_n)\cdot Out(p_1,f_{n+1})\cdot\ldots\cdot Out(p_n,f_{2n})\rangle$	$\frac{2}{n+2}+\frac{2n}{2n+2}$	n	
	Π^{f}_{e,f_n}	$\langle Move(f_0,f_n)\cdot In(p_n,f_n)\cdot Move(f_n,f_1)\cdot In(p_1,f_1)\cdot\ldots\cdot In(p_{n-1},f_{n-1})\cdot Out(p_1,f_{n+1})\cdot\ldots\cdot Out(p_n,f_{2n})\rangle$	$\frac{2}{n+2}+\frac{2n}{2n+2}$		
	$\Pi^{\mathsf{f}}_{e,f_{n+1}}$	$\langle In(p_1,f_1)\cdot\ldots\cdot In(p_n,f_n)\cdot Out(p_2,f_{n+2})\cdot\ldots\cdot Out(p_n,f_{2n})\cdot Move(f_0,f_{n+1})\cdot Out(p_1,f_{n+1})\rangle$	$\frac{1}{n+2}+\frac{2n}{2n+2}$	n	
	$\Pi^{\mathsf{f}}_{e,f_{2n}}$	$\langle In(p_1,f_1)\cdot\ldots\cdot In(p_n,f_n)\cdot Out(p_1,f_{n+1})\cdot\ldots\cdot Out(p_{n-1},f_{2n-1})\cdot Move(f_0,f_{2n})\cdot Out(p_n,f_{2n})\rangle$	$\frac{1}{n+2}+\frac{2n}{2n+2}$		
	$\Pi^{\mathsf{i}}_{p_i,1}$	$\langle Move(f_0,f_i)\cdot In(p_i,f_i)\cdot Move(f_i,f_{n+i})\cdot Out(p_i,f_{n+i})\rangle$	$\frac{2}{n+2}+\frac{2}{2n+2}$	n	

while an optimal plan for the original task, $\langle Move(f_0, f_1) \cdot In(p_1, f_1) \cdot Move(f_1, f_2) \cdot In(p_2, f_2) \cdot Move(f_2, f_3) \cdot \ldots \cdot In(p_n, f_n) \cdot Move(f_n, f_{n+1}) \cdot Out(p_1, f_{n+1}) \cdot Move(f_{n+1}, f_{n+2}) \cdot Out(p_2, f_{n+2}) \cdot Move(f_{n+2}, f_{n+3}) \cdot \ldots \cdot Out(p_n, f_{2n}) \rangle$, has the cost of $4n$, providing us with the upper bound of $1/2$ for the $h^{\mathcal{J}}$ and $h^{\mathcal{FJ}}$ heuristics in MICONIC. Putting this upper bound together with the previously obtained lower bound of $1/2$, we conclude that the asymptotic performance ratio of $h^{\mathcal{J}}$ and $h^{\mathcal{FJ}}$ in MICONIC is $1/2$.

Satellite

The SATELLITE domain is quite complex. A SATELLITE tasks consists of some satellites $\mathcal{S}$, each $s \in \mathcal{S}$ with a finite set of instruments $\mathcal{I}_s$ onboard, $\mathcal{I} = \bigcup_{s \in \mathcal{S}} \mathcal{I}_s$. There is a set of image modes $\mathcal{M}$, and for each mode $m \in \mathcal{M}$, there is a set $\mathcal{I}_m \subseteq \mathcal{I}$ of instruments supporting mode m. Likewise, there is a set of directions $\mathcal{L}$, image objectives $\mathcal{O} \subseteq \mathcal{L} \times \mathcal{M}$, and functions $cal : \mathcal{I} \mapsto \mathcal{L}$, $p_0 : \mathcal{S} \mapsto \mathcal{L}$, and $p_* : \mathcal{S}_0 \mapsto \mathcal{L}$ with $\mathcal{S}_0 \subseteq \mathcal{S}$, where cal is the calibration target direction function, p_0 is the initial direction function, and p_* is the goal pointing direction function.

Let us denote by $\mathcal{O}_i = \{o = (d, m) \in \mathcal{O} \mid i \in \mathcal{I}_m\}$ the subset of all images that can be taken by instrument i, by $\mathcal{O}^s = \bigcup_{i \in \mathcal{I}_s} \mathcal{O}_i$ the subset of all images that can be taken by instruments on satellite s, and by $\mathcal{S}_m = \{s \mid \mathcal{I}_s \cap \mathcal{I}_m \neq \emptyset\}$ the subset of all satellites that can take images in mode m. The problem description in SAS$^+$ is as follows.

- Variables $V = \mathcal{S} \cup \{On_i, C_i \mid i \in \mathcal{I}\} \cup \mathcal{O}$ with domains

$$\begin{aligned} \forall s \in \mathcal{S} :\ & \mathcal{D}(s) = \mathcal{L}, \\ \forall i \in \mathcal{I} :\ & \mathcal{D}(On_i) = \mathcal{D}(C_i) = \{0, 1\}, \\ \forall o \in \mathcal{O} :\ & \mathcal{D}(o) = \{0, 1\}. \end{aligned}$$

- Initial state $I = \{s = p_0(s) \mid s \in \mathcal{S}\} \cup \{On_i = 0, C_i = 0 \mid i \in \mathcal{I}\} \cup \{o = 0 \mid o \in \mathcal{O}\}$.

- Goal $G = \{s = p_*(s) \mid s \in \mathcal{S}_0\} \cup \{o = 1 \mid o \in \mathcal{O}\}$.

- Actions

$$\begin{aligned} A = \bigcup_{s \in \mathcal{S}} \big(& \{Turn(s, d, d') \mid \{d, d'\} \subseteq \mathcal{L}\} \cup \{SwOn(i, s), Cal(i, s), SwOff(i) \mid i \in \mathcal{I}_s\}\big) \cup \\ & \{TakeIm(o, d, s, i) \mid o = (d, m) \in \mathcal{O}, s \in \mathcal{S}_m, i \in \mathcal{I}_m \cap \mathcal{I}_s\}, \end{aligned}$$

  where

  - turn satellite: $Turn(s, d, d') = \langle \{s = d\}, \{s = d'\} \rangle$,
  - power on instrument: $SwOn(i, s) = \langle \{On_{i'} = 0 \mid i' \in \mathcal{I}_s\}, \{On_i = 1\} \rangle$,
  - power off instrument: $SwOff(i) = \langle \{On_i = 1\}, \{On_i = 0, C_i = 0\} \rangle$,
  - calibrate instrument: $Cal(i, s) = \langle \{C_i = 0, On_i = 1, s = cal(i)\}, \{C_i = 1\} \rangle$, and
  - take an image: $TakeIm(o, d, s, i) = \langle \{o = 0, C_i = 1, s = d\}, \{o = 1\} \rangle$.

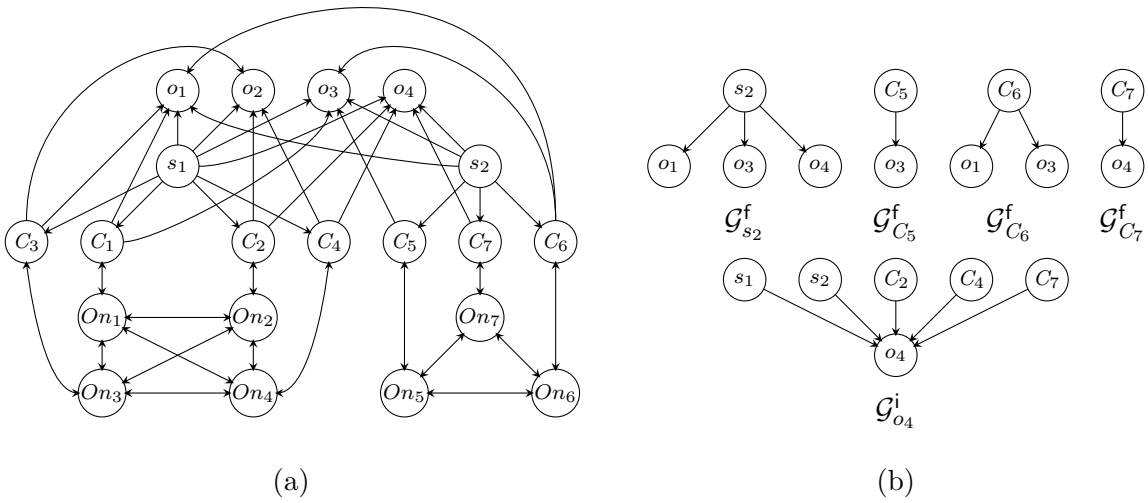


Figure 4.12: SATELLITE example task (a) causal graph and (b) a representative subset of the collection of v-forks and v-iforks

Fork Decomposition The causal graph of an example SATELLITE task and a representative subset of the collection of v-forks and v-iforks are depicted in Figure 4.12. Since the variables $\{On_i, C_i \mid i \in \mathcal{I}\} \cup \mathcal{S} \setminus \mathcal{S}_0$ have no goal value, the collection of v-forks and v-iforks will be as follows in the general case.

- For each satellite $s \in \mathcal{S}$, an s-fork with the leaves $\mathcal{O}^s$.
- For each instrument $i \in \mathcal{I}$, a C_i-fork with the leaves $\mathcal{O}_i$.
- For each image objective $o = (d, m) \in \mathcal{O}$, a o-ifork with the parents $\{C_i \mid i \in \mathcal{I}_m\} \cup \mathcal{S}_m$.

The root domains of all forks rooted at instruments $i \in \mathcal{I}$ and of all the inverted-fork sinks are binary in the first place, and the root domains of the forks rooted at satellites $s \in \mathcal{S}$ are abstracted as in Eq. 4.24 ("leave-one-out"). This provides us with

$$\begin{aligned}
\mathbf{\Pi}_{\mathcal{F}} &= \{\Pi^{\mathsf{f}}_{s,d} \mid s \in \mathcal{S}, d \in \mathcal{L}\} \cup \{\Pi^{\mathsf{f}}_{C_i} \mid i \in \mathcal{I}\}, \\
\mathbf{\Pi}_{\mathcal{I}} &= \{\Pi^{\mathsf{i}}_{o} \mid o \in \mathcal{O}\}, \\
\mathbf{\Pi}_{\mathcal{FI}} &= \{\Pi^{\mathsf{f}}_{s,d} \mid s \in \mathcal{S}, d \in \mathcal{L}\} \cup \{\Pi^{\mathsf{f}}_{C_i} \mid i \in \mathcal{I}\} \cup \{\Pi^{\mathsf{i}}_{o} \mid o \in \mathcal{O}\}.
\end{aligned}$$

The total number of forks is thus $n^{\mathsf{f}} = |\mathcal{S}| \cdot |\mathcal{L}| + |\mathcal{I}|$ and the total number of inverted forks is $n^{\mathsf{i}} = |\mathcal{O}|$. For each action $a \in A$, the number of its representatives in each abstract task, as well as the cost assigned to each such representative, are given in Figure 4.13.

Lower Bounds First, note that any optimal plan for a SATELLITE task contains at most 6 actions per image objective $o \in \mathcal{O}$ and one action per satellite $s \in \mathcal{S}_0$ such that $I[s] \neq G[s]$. Now we show that each of the three heuristics fully account for the cost of at least one action per image objective $o \in \mathcal{O}$ and one action per such a satellite. This will provide us with the lower bound of $1/6$ on the asymptotic performance ratios of our three heuristics.

Lemma 4 *For any* SATELLITE *task, $h^{\mathcal{F}}$, $h^{\mathcal{I}}$, and $h^{\mathcal{FI}}$ fully account for the cost of at least one* **Take Image** *action $TakeIm(o, d, s, i)$ for each image objective $o \in \mathcal{O}$.*

							$o \in \mathcal{O}_i$	$o \in \mathcal{O}^s \setminus \mathcal{O}_i$	$o \notin \mathcal{O}^s$			
Action	$\Pi^{\mathsf{f}}_{s,d}$	$\Pi^{\mathsf{f}}_{s,d'}$	$\Pi^{\mathsf{f}}_{s,d''}$	$\Pi^{\mathsf{f}}_{s',d^*}$	$\Pi^{\mathsf{f}}_{C_i}$	$\Pi^{\mathsf{f}}_{C_{i'}}$	Π^{i}_{o}	Π^{i}_{o}	Π^{i}_{o}	$\mathbf{\Pi}_{\mathcal{F}}$	$\mathbf{\Pi}_{\mathcal{J}}$	$\mathbf{\Pi}_{\mathcal{FJ}}$
$Turn(s,d,d')$	1	1	0	0	0	0	1	1	0	$\frac{1}{2}$	$\frac{1}{\lvert\mathcal{O}^s\rvert}$	$\frac{1}{\lvert\mathcal{O}^s\rvert+2}$
$SwOn(i,s)$	0	0	0	0	0	0	0	0	0	0	0	0
$Cal(i,s)$	0	0	0	0	1	0	1	0	0	1	$\frac{1}{\lvert\mathcal{O}_i\rvert}$	$\frac{1}{\lvert\mathcal{O}_i\rvert+1}$
$SwOff(i)$	0	0	0	0	1	0	1	0	0	1	$\frac{1}{\lvert\mathcal{O}_i\rvert}$	$\frac{1}{\lvert\mathcal{O}_i\rvert+1}$

(a)

	$s' \in \mathcal{S}_m$	$s' \notin \mathcal{S}_m$	$i' \in \mathcal{I}_m$	$i' \notin \mathcal{I}_m$					
Action	$\Pi^{\mathsf{f}}_{s',d'}$	$\Pi^{\mathsf{f}}_{s',d'}$	$\Pi^{\mathsf{f}}_{C_{i'}}$	$\Pi^{\mathsf{f}}_{C_{i'}}$	Π^{i}_{o}	$\Pi^{\mathsf{i}}_{o'}$	$\mathbf{\Pi}_{\mathcal{F}}$	$\mathbf{\Pi}_{\mathcal{J}}$	$\mathbf{\Pi}_{\mathcal{FJ}}$
$TakeIm(o,d,s,i)$, $o=(d,m)$	1	0	1	0	1	0	$\frac{1}{\lvert\mathcal{S}_m\rvert\cdot\lvert\mathcal{L}\rvert+\lvert\mathcal{I}_m\rvert}$	1	$\frac{1}{\lvert\mathcal{S}_m\rvert\cdot\lvert\mathcal{L}\rvert+\lvert\mathcal{I}_m\rvert+1}$

(b)

Figure 4.13: Number of representatives for each original SATELLITE action in each abstract task, as well as the partition of the action costs between these representatives; table (a) shows **Turn**, **Switch On**, **Switch Off**, and **Calibrate** actions, and table (b) shows **Take Image** actions

Proof: For an image objective $o = (d, m) \in \mathcal{O}$, some actions $TakeIm(o, d, s, i) = \langle\{o = 0, C_i = 1, s = d\}, \{o = 1\}\rangle$ will appear in optimal plans for $|\mathcal{S}_m| \cdot |\mathcal{L}|$ fork abstract tasks rooted in satellites, $|\mathcal{I}_m|$ fork abstract tasks rooted in instrument calibration status variables C_i, and one inverted-fork abstract task with sink o. Together with the costs of the action representatives in the abstract problems (see Figure 4.13), we have

$h^{\mathcal{F}}$: cost of each representative is $\frac{1}{|\mathcal{S}_m|\cdot|\mathcal{L}|+|\mathcal{I}_m|}$ and there are $|\mathcal{S}_m| \cdot |\mathcal{L}| + |\mathcal{I}_m|$ fork abstract tasks,

$h^{\mathcal{J}}$: cost of each representative is 1 and there is one inverted fork abstract task, and

$h^{\mathcal{FJ}}$: cost of each representative is $\frac{1}{|\mathcal{S}_m|\cdot|\mathcal{L}|+|\mathcal{I}_m|+1}$ and there are $|\mathcal{S}_m| \cdot |\mathcal{L}| + |\mathcal{I}_m| + 1$ abstract tasks.

Therefore, for each $o \in \mathcal{O}$, the cost of one $TakeIm(o, d, s, i)$ action will be fully accounted for by each of the three heuristics. ∎

Lemma 5 *For any* SATELLITE *task, $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$ fully account for the cost of at least one* **Turn** *action $Turn(s, d, d')$ for each $s \in \mathcal{S}_0$ such that $I[s] \neq G[s]$.*

Proof: If $s \in \mathcal{S}_0$ is a satellite with $I[s] \neq G[s]$, then an action $Turn(s, I[s], d')$ will appear in any optimal plan for $\Pi^{\mathsf{f}}_{s,I[s]}$, an action $Turn(s, d, G[s])$ will appear in any optimal plan for $\Pi^{\mathsf{f}}_{s,G[s]}$, and for each $o \in \mathcal{O}^s$, an action $Turn(s, d, G[s])$ will appear in any optimal plan for Π^{i}_{o}. Together with the costs of the action representatives in the abstract problems (see Figure 4.13) we have

$h^{\mathcal{F}}$: cost of each representative is $\frac{1}{2}$ and there are 2 fork abstract tasks,

$h^{\mathcal{J}}$: cost of each representative is $\frac{1}{|\mathcal{O}^s|}$ and there are $|\mathcal{O}^s|$ inverted fork abstract tasks, and

$h^{\mathcal{FJ}}$: cost of each representative is $\frac{1}{|\mathcal{O}^s|+2}$ and there are $|\mathcal{O}^s| + 2$ abstract tasks.

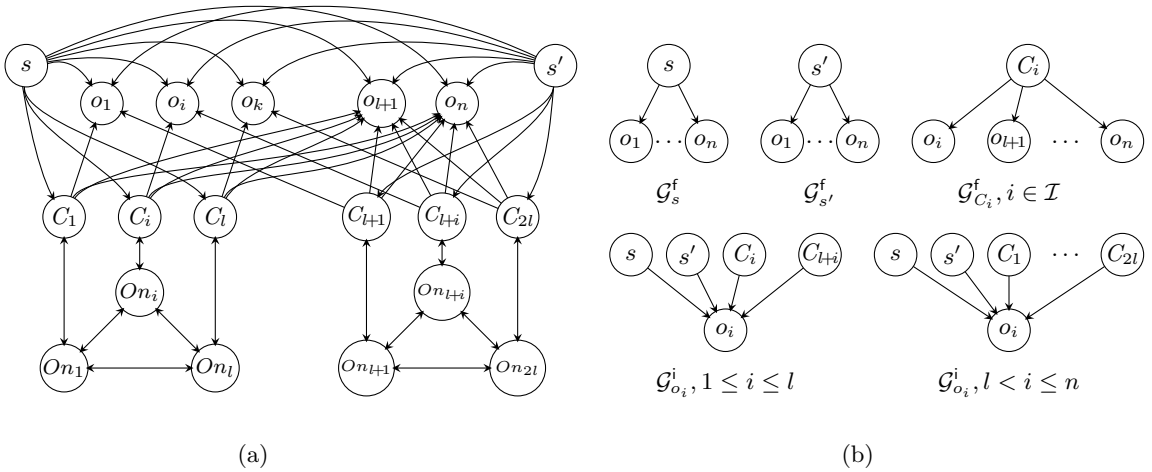


Figure 4.14: (a) Causal graph and (b) the corresponding collection of v-forks and v-iforks for the SATELLITE task used in the proof of the upper bound of 1/6

Therefore, for each $s \in \mathcal{S}_0$ such that $I[s] \neq G[s]$, the cost of one $Turn(s, d, d')$ action will be fully accounted for by each of the three heuristics. ■

Together, Lemmas 4 and 5 imply that, for $h \in \{h^{\mathcal{F}}, h^{\mathcal{J}}, h^{\mathcal{FJ}}\}$, on SATELLITE we have $\frac{h}{h^*} \geq 1/6$.

Upper Bound A SATELLITE task on which all three heuristics achieve the ratio of exactly 1/6 consists of two identical satellites $\mathcal{S} = \{s, s'\}$ with l instruments each, $\mathcal{I} = \mathcal{I}_s \cup \mathcal{I}_{s'} = \{1, \ldots, l\} \cup \{l+1, \ldots, 2l\}$, such that instruments $\{i, l+i\}$ have two modes each: m_0 and m_i. There is a set of $n+1$ directions $\mathcal{L} = \{d_I, d_1, \ldots, d_n\}$ and a set of n image objectives $\mathcal{O} = \{o_1, \ldots, o_n\}, o_i = (d_I, m_i)$ for $1 \leq i \leq l$ and $o_i = (d_i, m_0)$ for $l < i \leq n$. The calibration direction of instruments $\{i, l+i\}$ is d_i. The SAS$^+$ encoding for this planning task is as follows.

- Variables $V = \mathcal{S} \cup \mathcal{O} \cup \{On_i, C_i \mid i \in \mathcal{I}\}$.
- Initial state $I = \{s = d_I \mid s \in \mathcal{S}\} \cup \{On_i = 0, C_i = 0 \mid i \in \mathcal{I}\} \cup \{o = 0 \mid o \in \mathcal{O}\}$.
- Goal $G = \{o = 1 \mid o \in \mathcal{O}\}$.
- Actions

$$A = \bigcup_{s \in \mathcal{S}} \left(\{Turn(s, d, d') \mid \{d, d'\} \subseteq \mathcal{L}\} \cup \{SwOn(i, s), Cal(i, s), SwOff(i) \mid i \in \mathcal{I}_s\} \right) \cup$$
$$\bigcup_{s \in \mathcal{S}} \left(\{TakeIm((d_I, m_i), d_I, s, i) \mid i \in \mathcal{I}_s\} \cup \bigcup_{j=l+1}^{n} \{TakeIm((d_j, m_0), d_j, s, i) \mid i \in \mathcal{I}_s\} \right).$$

The causal graph of this task is depicted in Figure 4.14a. The state variables $\{On_i, C_i \mid i \in \mathcal{I}\} \cup \mathcal{S}$ have no goal value, and thus the collection of v-forks and v-iforks for this task is as in Figure 4.14b. The domains of the inverted-fork sinks are binary, and the domains of the fork roots are abstracted

h	task	optimal plan	cost	#	$h(I)$
$h^{\mathcal{F}}$	$\Pi^{\mathsf{f}}_{s,d}$	$\langle TakeIm(o_1,d_I,s',l+1)\cdot\ldots\cdot TakeIm(o_l,d_I,s',2l)\cdot TakeIm(o_{l+1},d_{l+1},s',2l)\cdot\ldots\cdot TakeIm(o_n,d_n,s',2l)\rangle$	$\frac{l}{2n+4}+\frac{n-l}{2n+2l+2}$	$n+1$	n
	$\Pi^{\mathsf{f}}_{s',d}$	$\langle TakeIm(o_1,d_I,s,1)\cdot\ldots\cdot TakeIm(o_l,d_I,s,l)\cdot TakeIm(o_{l+1},d_{l+1},s,l)\cdot\ldots\cdot TakeIm(o_n,d_n,s,l)\rangle$	$\frac{l}{2n+4}+\frac{n-l}{2n+2l+2}$	$n+1$	
	$\Pi^{\mathsf{f}}_{C_i}, i\in\mathcal{I}_s$	$\langle TakeIm(o_i,d_I,s',l+i)\cdot TakeIm(o_{l+1},d_{l+1},s',2l)\cdot\ldots\cdot TakeIm(o_n,d_n,s',2l)\rangle$	$\frac{1}{2n+4}+\frac{n-l}{2n+2l+2}$	l	
	$\Pi^{\mathsf{f}}_{C_i}, i\in\mathcal{I}_{s'}$	$\langle TakeIm(o_i,d_I,s,i)\cdot TakeIm(o_{l+1},d_{l+1},s,l)\cdot\ldots\cdot TakeIm(o_n,d_n,s,l)\rangle$	$\frac{1}{2n+4}+\frac{n-l}{2n+2l+2}$	l	
$h^{\mathcal{J}}$	$\Pi^{\mathsf{i}}_{o_j}, 1\le j\le l$	$\langle Turn(s,d_I,d_j)\cdot Cal(j,s)\cdot Turn(s,d_j,d_I)\cdot TakeIm(o_j,d_I,s,j)\rangle$	$\frac{2}{n}+\frac{1}{n-l+1}+1$	l	$n+2+\frac{n}{n-l+1}$
	$\Pi^{\mathsf{i}}_{o_j}, l<j\le n$	$\langle Turn(s,d_I,d_1)\cdot Cal(1,s)\cdot Turn(s,d_1,d_j)\cdot TakeIm(o_j,d_I,s,1)\rangle$	$\frac{2}{n}+\frac{1}{n-l+1}+1$	$n-l$	
$h^{\mathcal{FJ}}$	$\Pi^{\mathsf{f}}_{s,d}$	$\langle TakeIm(o_1,d_I,s',l+1)\cdot\ldots\cdot TakeIm(o_l,d_I,s',2l)\cdot TakeIm(o_{l+1},d_{l+1},s',2l)\cdot\ldots\cdot TakeIm(o_n,d_n,s',2l)\rangle$	$\frac{l}{2n+5}+\frac{n-l}{2n+2l+3}$	$n+1$	$n+\frac{2n}{n+2}+\frac{n}{n-l+2}$
	$\Pi^{\mathsf{f}}_{s',d}$	$\langle TakeIm(o_1,d_I,s,1)\cdot\ldots\cdot TakeIm(o_l,d_I,s,l)\cdot TakeIm(o_{l+1},d_{l+1},s,l)\cdot\ldots\cdot TakeIm(o_n,d_n,s,l)\rangle$	$\frac{l}{2n+5}+\frac{n-l}{2n+2l+3}$	$n+1$	
	$\Pi^{\mathsf{f}}_{C_i}, i\in\mathcal{I}_s$	$\langle TakeIm(o_i,d_I,s',l+i)\cdot TakeIm(o_{l+1},d_{l+1},s',2l)\cdot\ldots\cdot TakeIm(o_n,d_n,s',2l)\rangle$	$\frac{1}{2n+5}+\frac{n-l}{2n+2l+3}$	l	
	$\Pi^{\mathsf{f}}_{C_i}, i\in\mathcal{I}_{s'}$	$\langle TakeIm(o_i,d_I,s,i)\cdot TakeIm(o_{l+1},d_{l+1},s,l)\cdot\ldots\cdot TakeIm(o_n,d_n,s,l)\rangle$	$\frac{1}{2n+5}+\frac{n-l}{2n+2l+3}$	l	
	$\Pi^{\mathsf{i}}_{o_j}, 1\le j\le l$	$\langle Turn(s,d_I,d_j)\cdot Cal(j,s)\cdot Turn(s,d_j,d_I)\cdot TakeIm(o_j,d_I,s,j)\rangle$	$\frac{2}{n+2}+\frac{1}{n-l+2}+\frac{1}{2n+5}$	l	
	$\Pi^{\mathsf{i}}_{o_j}, l<j\le n$	$\langle Turn(s,d_I,d_1)\cdot Cal(1,s)\cdot Turn(s,d_1,d_j)\cdot TakeIm(o_j,d_I,s,1)\rangle$	$\frac{2}{n+2}+\frac{1}{n-l+2}+\frac{1}{2n+2l+3}$	$n-l$	

Table 4.5: Optimal plans for the abstract tasks and the overall heuristic estimates for the SATELLITE task used in the proof of the upper bound of 1/6

as in Eq. 4.24 ("leave-one-out"). This provides us with

$$\begin{aligned}\mathbf{\Pi}_{\mathcal{F}} &= \{\Pi^{\mathsf{f}}_{s,d}, \Pi^{\mathsf{f}}_{s',d} \mid d\in\mathcal{L}\}\cup\{\Pi^{\mathsf{f}}_{C_i} \mid i\in\mathcal{I}\},\\ \mathbf{\Pi}_{\mathcal{J}} &= \{\Pi^{\mathsf{i}}_{o} \mid o\in\mathcal{O}\},\\ \mathbf{\Pi}_{\mathcal{FJ}} &= \{\Pi^{\mathsf{f}}_{s,d}, \Pi^{\mathsf{f}}_{s',d} \mid d\in\mathcal{L}\}\cup\{\Pi^{\mathsf{f}}_{C_i} \mid i\in\mathcal{I}\}\cup\{\Pi^{\mathsf{i}}_{o} \mid o\in\mathcal{O}\}.\end{aligned}$$

The total number of forks in this task is $n^{\mathsf{f}} = 2n+2l+2$ and the total number of inverted forks is $n^{\mathsf{i}} = n$. The costs of the action representatives in each abstract task are given in Figure 4.13, where $|\mathcal{O}^s| = |\mathcal{O}^{s'}| = |\mathcal{O}| = n$, $|\mathcal{O}_i| = n-l+1$, $|\mathcal{S}_m| = 2$, $|\mathcal{I}_{m_0}| = 2l$, $|\mathcal{I}_{m_i}| = 2$, and $|\mathcal{L}| = n+1$.

The optimal plans per abstract task are depicted in Table 4.5, while an optimal plan for the original problem, $\langle SwOn(1,s)\cdot Turn(s,d_I,d_1)\cdot Cal(1,s)\cdot Turn(s,d_1,d_I)\cdot TakeIm(o_1,d_I,s,1)\cdot SwOff(1)\cdot\ldots SwOn(l-1,s)\cdot Turn(s,d_I,d_{l-1})\cdot Cal(l-1,s)\cdot Turn(s,d_{l-1},d_I)\cdot TakeIm(o_{l-1},d_I,s,l-1)\cdot SwOff(l-1)\cdot SwOn(l,s)\cdot Turn(s,d_I,d_l)\cdot Cal(l,s)\cdot Turn(s,d_l,d_I)\cdot TakeIm(o_l,d_I,s,l)\cdot Turn(s,d_I,d_{l+1})\cdot TakeIm(o_{l+1},d_{l+1},s,l)\cdot\ldots\cdot Turn(s,d_{n-1},d_n)\cdot TakeIm(o_n,d_n,s,l)\rangle$, has the cost of $4l+2n-1$. For $l=n-\sqrt{n}$, this provides us with the asymptotic performance ratio of 1/6 for all three heuristics.

4.4.2 Experimental Evaluation

To evaluate the practical attractiveness of the fork-decomposition heuristics, we have conducted an empirical study on a wide sample of planning domains from the International Planning Competitions (IPC) 1998-2006, plus a non-IPC SCHEDULE-STRIPS domain.[5] The domains were selected to

[5]SCHEDULE-STRIPS appears in the domains' distribution of IPC-2000. Later we became aware of the fact that this domain was excluded from the competition because its encoding generated problems for various planners.

domain	S	$h^{\mathcal{F}}$		$h^{\mathcal{J}}$		$h^{\mathcal{FJ}}$		MS-10^4		MS-10^5		HSP^*_{F}		Gamer		blind		$h_{\max}$	
		s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$
airport-ipc4	21	11	52	14	67	11	52	19	90	17	81	15	71	11	52	18	86	20	95
blocks-ipc2	30	17	57	15	50	15	50	18	60	20	67	30	100	30	100	18	60	18	60
depots-ipc3	7	2	29	2	29	2	29	7	100	4	57	4	57	4	57	4	57	4	57
driverlog-ipc3	12	9	75	10	83	9	75	12	100	12	100	9	75	11	92	7	58	8	67
freecell-ipc3	5	3	60	2	40	2	40	5	100	1	20	5	100	2	40	4	80	5	100
grid-ipc1	2	1	50	1	50	1	50	2	100	2	100	0	0	2	100	1	50	2	100
gripper-ipc1	20	5	25	5	25	5	25	7	35	7	35	6	30	20	100	7	35	7	35
logistics-ipc1	6	3	50	2	33	2	33	4	67	5	83	3	50	6	100	2	33	2	33
logistics-ipc2	22	21	95	15	68	14	64	16	73	21	95	16	73	20	91	10	45	10	45
miconic-strips-ipc2	85	45	53	42	49	40	47	54	64	55	65	45	53	85	100	50	59	50	59
mprime-ipc1	24	17	71	17	71	17	71	21	88	12	50	8	33	9	38	19	79	24	100
mystery-ipc1	21	16	76	15	71	16	76	17	81	13	62	11	52	8	38	18	86	18	86
openstacks-ipc5	7	7	100	7	100	7	100	7	100	7	100	7	100	7	100	7	100	7	100
pathways-ipc5	4	4	100	4	100	4	100	3	75	4	100	4	100	4	100	4	100	4	100
pipes-notank-ipc4	21	9	43	11	52	8	38	20	95	12	57	13	62	11	52	14	67	17	81
pipes-tank-ipc4	14	6	43	6	43	6	43	13	93	7	50	7	50	6	43	10	71	10	71
psr-small-ipc4	50	47	94	48	96	47	94	50	100	50	100	50	100	47	94	48	96	49	98
rovers-ipc5	7	5	71	6	86	6	86	6	86	7	100	6	86	5	71	5	71	6	86
satellite-ipc4	6	6	100	6	100	5	83	6	100	6	100	5	83	6	100	4	67	5	83
schedule-strips	43	42	98	35	81	39	91	22	51	1	2	11	26	3	7	29	67	31	72
tpp-ipc5	6	5	83	5	83	5	83	6	100	6	100	5	83	5	83	5	83	6	100
trucks-ipc5	9	5	56	5	56	5	56	6	67	5	56	9	100	3	33	5	56	7	78
zenotravel-ipc3	11	8	73	9	82	8	73	11	100	11	100	8	73	10	91	7	64	8	73
total	433	294		282		274		332		285		277		315		296		318	

Table 4.6: A summary of the experimental results. Per domain, S denotes the number of tasks solved by *any* planner. Per planner/domain, the number of tasks solved by that planner is given both by the absolute number (s) and by the percentage from "solved by some planners" ($\%S$). The last row summarize the number of solved instances.

allow a comparative evaluation with other, both baseline and state-of-the-art, approaches/planners, not all of which supported all the PDDL features at the time of our evaluation.

Later we formally prove that, under ad hoc action cost partitions such as our uniform partition, none of the three fork decompositions as in Definition 14 is dominated by the other two. Hence, we have implemented three additive fork-decomposition heuristics, $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$, within the standard heuristic forward search framework of the Fast Downward planner (Helmert, 2006) using the A^* algorithm with full duplicate elimination. The $h^{\mathcal{F}}$ heuristic corresponds to the ensemble of all (not clearly redundant) fork subgraphs of the causal graph, with the domains of the roots being abstracted using the "leave-one-value-out" binary-valued domain decompositions as follows:

$$\forall \vartheta_i \in \mathcal{D}(v): \quad \phi^{\mathsf{f}}_{v,i}(\vartheta) = \begin{cases} 0, & \vartheta = \vartheta_i \\ 1, & \text{otherwise} \end{cases} . \tag{4.24}$$

The $h^{\mathcal{J}}$ heuristic is the same but for the inverted fork subgraphs, with the domains of the sinks being abstracted using the "distance-to-goal-value" ternary-valued domain decompositions[6] as in Eq. 4.25.

$$\forall \vartheta \in \mathcal{D}(v): \quad \phi^{\mathsf{i}}_{v,i}(\vartheta) = \begin{cases} 0, & d(\vartheta, G[v]) < 2i-1 \\ 1, & d(\vartheta, G[v]) = 2i-1 \\ 2, & d(\vartheta, G[v]) > 2i-1 \end{cases} . \tag{4.25}$$

The ensemble of the $h^{\mathcal{FJ}}$ heuristic is the union of these for $h^{\mathcal{F}}$ and $h^{\mathcal{J}}$. The action cost partition in all three heuristics was what we call "uniform."

[6]While "distance-from-initial-value" is reasonable for the evaluation of just the initial state, "leave-one-value-out" for fork roots and "distance-to-goal-value" for inverted-fork sinks should typically be much more attractive for the evaluation of all the states examined by A^*.

We make a comparison with two baseline approaches, namely "blind A^*" with heuristic value 0 for goal states and 1 otherwise, and A^* with the $h_{\max}$ heuristic (Bonet & Geffner, 2001), as well as with state-of-the-art abstraction heuristics, represented by the merge-and-shrink abstractions of Helmert et al. (2007). The latter were constructed under the linear, f-preserving abstraction strategy proposed by these authors, and this under two fixed bounds on the size of the abstract state spaces, notably $|S^\alpha| < 10^4$ and $|S^\alpha| < 10^5$. These four (baseline and merge-and-shrink) heuristics were implemented by Helmert et al. (2007) within the same planning system as our fork-decomposition heuristics, allowing for a fairly unbiased comparison. We also compare to the Gamer (Edelkamp & Kissmann, 2009) and HSP^*_F (Haslum, 2008) planners, the winner and the runner-up at the sequential optimization track of IPC-2008. On the algorithmic side, Gamer is based on a bidirectional blind search using sophisticated symbolic-search techniques, and HSP^*_F uses A^* with an additive critical-path heuristic. The experiments were conducted on a 3 GHz Intel E8400 CPU with 2 GB memory, using 1.5 GB memory limit and 30 minute timeout. The only exception was Gamer, for which we used similar machines but with 4 GB memory and 2 GB memory limit; this was done to provide Gamer with the environment for which it was configured.

Table 4.6 summarizes our experimental results in terms of the number of tasks solved by each planner. Our impression of fork-decomposition heuristics from Table 4.6 is somewhat mixed. On the one hand, the performance of all three fork-decomposition based planners was comparable to one of the settings of the merge-and-shrink heuristic, and this clearly testifies for that the framework of implicit abstractions is not of theoretical interest only. On the other hand, all the planners, except for A^* with the merge-and-shrink heuristic with $|S^\alpha| < 10^4$, failed to outperform A^* with the baseline $h_{\max}$ heuristic. More important for us is that, unfortunately, all three fork-decomposition based planners failed to outperform even the basic blind search.

This, however, is not the end of the story for the fork-decomposition heuristics. Some hope can be found in the detailed results in Tables B.1-B.6 (pp. 177-182). As it appears from Table B.2, on, e.g., the LOGISTICS-IPC2 domain, $h^\mathcal{F}$ almost consistently leads to expanding fewer search nodes than the (better between the two merge-and-shrink heuristics on this domain) MS-10^5, with the difference hitting four orders of magnitude. However, the time complexity of $h^\mathcal{F}$ per search node is substantially higher than that of MS-10^5, with the two expanding at a rate of approximately 40 and 100000 nodes per second, respectively. The outcome is simple: while with no time limits (and only memory limit of 1.5 GB) $h^\mathcal{F}$ solves more tasks in LOGISTICS-IPC2 than MS-10^5 (task 12-1 is solved with $h^\mathcal{F}$ in 2519.01 seconds), this is not so with a standard time limit of half an hour used for Table B.2. In what follows we examine the possibility of exploiting the informativeness of fork-decomposition heuristics while not falling into the trap of costly per-node heuristic evaluation.

4.5 Back to Theory: h-Partitions and Databased Implicit Abstractions

Accuracy and low time complexity are both desired yet competing properties of heuristic functions. For many powerful heuristics, and abstraction heuristics in particular, computing $h(s)$ for each state s in isolation is impractical: while computing $h(s)$ is polynomial in the description size of Π, it is often not efficient enough to be performed at each search node. However, for some costly heuristics this obstacle can be largely overcome by sharing most of the computation between the evaluations of h on different states. If that is possible, the shared parts of computing h for *all* problem states are precomputed and memorized before the search, and then reused during the search by the evaluations

of h on different states. Such a mixed offline/online heuristic computation is henceforth called **h-partition**, and we define the time complexity of an h-partition as the complexity of computing h for a *set* of states. Given a subset of k problem states $S' \subseteq S$, the h-partition's time complexity of computing $\{h(s) \mid s \in S'\}$ is expressed as $O(X + kY)$, where $O(X)$ and $O(Y)$ are, respectively, the complexity of the (offline) pre-search and (online) per-node parts of computing $h(s)$.

These days h-partitions are being adopted by various optimal planners using critical-path heuristics h^m for $m > 1$ (Haslum et al., 2005), landmark heuristics h^L and h^{LA} (Karpas & Domshlak, 2009), and PDB and merge-and-shrink abstraction heuristics (Edelkamp, 2001; Helmert et al., 2007). Without effective h-partitions, optimal search with these heuristics would not scale up well, while with such h-partitions it constitutes the state of the art of cost-optimal planning. For instance, a very attractive property of PDB abstractions is the complexity of their natural h^α-partition. Instead of computing $h^\alpha(s) = h^*(\alpha(s))$ from scratch for each evaluated state s (impractical for all but tiny projections), the practice is to precompute and store $h^*(s')$ for *all* abstract states $s' \in S_\alpha$, after which the per-node computation of $h^\alpha(s)$ boils down to a hash-table lookup for $h^*(\alpha(s))$ with a perfect hash function. In our terms, the time and space complexity of that PDB h^α-partition for a set of k states is $O(|S^\alpha|(\log(|S^\alpha|) + |A|) + k)$ and $O(|S^\alpha|)$, respectively. This is precisely what makes PDB heuristics so attractive in practice. In that respect, the picture with merge-and-shrink abstractions is very much similar. While the order in which composites are formed and the choice of abstract states to contract are crucial to the complexity of their natural h^α-partitions, the time and space complexity for the concrete linear abstraction strategy of Helmert *et al.* are respectively $O(|V||S^\alpha|(\log(|S^\alpha|) + |A|) + k \cdot |V|)$ and $O(|S^\alpha|)$. Similarly to PDB abstractions, the per-node computation of $h^\alpha(s)$ with a merge-and-shrink abstraction α is just a lookup in a data structure storing $h^*(\alpha(s))$ for all abstract states $\alpha(s) \in S_\alpha$. Hence, while the pre-search computation with MS abstractions can be more costly than with PDBs, the online part of computing heuristic values is still extremely efficient. This per-node efficiency provides the merge-and-shrink heuristics with impressive practical effectiveness on numerous IPC domains (Helmert et al., 2007).

To sum up, we can say that the fixed size of abstract spaces induced by explicit abstractions such as PDBs and merge-and-shrink is not only a limitation but also a key to obtaining effective h-partitions. In contrast, escaping that limitation with implicit abstractions might trap us into having to pay a high price for each search-node evaluation. We now show, however, that the time-per-node complexity bottleneck of fork-decomposition heuristics can be successfully overcome. Specifically, we show that an equivalent of PDB's and merge-and-shrink notion of "database" exists for fork-decomposition abstractions as well, despite their exponential-size abstract spaces. Of course, unlike with PDB and merge-and-shrink abstractions, the *databased fork-decomposition heuristics* do not (and cannot) provide us with a purely lookup online computation of $h^\alpha(s)$. The online part of the h^α-partition has to be nontrivial in the sense that its complexity cannot be $O(1)$. In what comes next we prove the existence of such effective h-partitions for fork and inverted fork abstractions. In Section 4.5.3 we then empirically show that these h-partitions lead to fast pre-search and per-node computations, allowing the informativeness of the fork-decomposition heuristics to be successfully exploited in practice.

4.5.1 Fork Databases

Theorem 11 *Let $\Pi = \langle V, A, I, G, cost\rangle$ be a planning task with a fork causal graph rooted at a binary-valued variable r. There exists an h^*-partition for Π such that, for any set of k states, the time and space complexity of that h^*-partition is, respectively, $O(d^3|V|+|A_r|+kd|V|)$ and $O(d^2|V|)$,*

where $d = \max_v \mathcal{D}(v)$.

Proof: The proof is by a modification of the polynomial-time algorithm for computing $h^*(s)$ for a state s of such a task Π used in the proof of Theorem 4 (Tractable Forks). *Given a state* s, let $\mathcal{D}(r) = \{0, 1\}$, where $s[r] = 0$. In what follows, for each of the two root's values $\vartheta \in \mathcal{D}(r)$, $\neg\vartheta$ denotes the opposite value $1 - \vartheta$; $\sigma(r)$, $\trianglerighteq^*[\sigma(r)]$, DTG_v^0 and DTG_v^1 are defined exactly as in the proof of Theorem 4.

(1) For each of the two values $\vartheta_r \in \mathcal{D}(r)$ of the root variable, each leaf variable $v \in V \setminus \{r\}$, and each pair of values $\vartheta, \vartheta' \in \mathcal{D}(v)$, let $p_{\vartheta,\vartheta';\vartheta_r}$ be the cost of the cheapest sequence of actions changing v from ϑ to ϑ' *provided* $r = \vartheta_r$. The whole set $\{p_{\vartheta,\vartheta';\vartheta_r}\}$ for all the leaves $v \in V \setminus \{r\}$ can be computed by a straightforward variant of the all-pairs-shortest-paths, Floyd-Warshall algorithm on $DTG_v^{\vartheta_r}$ in time $O(d^3|V|)$.

(2) For each leaf variable $v \in V \setminus \{r\}$, $1 \leq i \leq d + 1$, and $\vartheta \in \mathcal{D}(v)$, let $g_{\vartheta;i}$ be the cost of the cheapest sequence of actions changing $s[v]$ to ϑ *provided* a sequence $\sigma \in \trianglerighteq^*[\sigma(r)]$, $|\sigma| = i$, of value changes of r. Having the values $\{p_{\vartheta,\vartheta';\vartheta_r}\}$ from step (1), the set $\{g_{\vartheta;i}\}$ is given by the solution of the recursive equation

$$g_{\vartheta;i} = \begin{cases} p_{s[v],\vartheta;s[r]}, & i = 1 \\ \min\limits_{\vartheta'} \left[g_{\vartheta';i-1} + p_{\vartheta',\vartheta;s[r]} \right], & 1 < i \leq \delta_\vartheta,\ i \text{ is odd} \\ \min\limits_{\vartheta'} \left[g_{\vartheta';i-1} + p_{\vartheta',\vartheta;\neg s[r]} \right], & 1 < i \leq \delta_\vartheta,\ i \text{ is even} \\ g_{\vartheta;i-1}, & \delta_\vartheta < i \leq d + 1 \end{cases},$$

where $\delta_\vartheta = |\mathcal{D}(v)| + 1$. Given that, we have

$$h^*(s) = \min_{\sigma \in \trianglerighteq^*[\sigma(r)]} \left[cost(\sigma) + \sum_{v \in V \setminus \{r\}} g_{G[v];|\sigma|} \right],$$

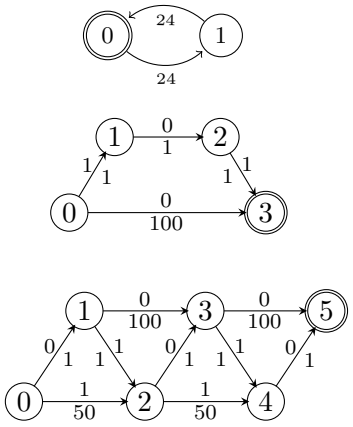
with $cost(\sigma) = \sum_{i=2}^{|\sigma|} cost(a_{\sigma[i]})$, where $a_{\sigma[i]} \in A$ is the cheapest action changing the value of r from $\sigma[i-1]$ to $\sigma[i]$.

Note that step (1) is already state-independent, but the heavy step (2) is not. However, the state dependence of step (2) can mostly be overcome as follows. For each $v \in V \setminus \{r\}$, $\vartheta \in \mathcal{D}(v)$, $1 \leq i \leq d + 1$, and $\vartheta_r \in \mathcal{D}(r)$, let $\tilde{g}_{\vartheta;i}(\vartheta_r)$ be the cost of the cheapest sequence of actions changing ϑ to $G[v]$ *provided* the value changes of r induce a 0/1 sequence of length i starting with ϑ_r. The set $\{\tilde{g}_{\vartheta;i}(\vartheta_r)\}$ is given by the solution of the recursive equation

$$\tilde{g}_{\vartheta;i}(\vartheta_r) = \begin{cases} p_{\vartheta,G[v];\vartheta_r}, & i = 1 \\ \min\limits_{\vartheta'} \left[\tilde{g}_{\vartheta';i-1}(\neg\vartheta_r) + p_{\vartheta,\vartheta';\vartheta_r} \right], & 1 < i \leq \delta_\vartheta \\ \tilde{g}_{\vartheta;i-1}(\vartheta_r), & \delta_\vartheta < i \leq d + 1 \end{cases}, \tag{4.26}$$

which can be solved in time $O(d^3|V|)$. Note that this equation is now independent of the evaluated state s, and yet $\{\tilde{g}_{\vartheta;i}(\vartheta_r)\}$ allow for computing $h^*(s)$ for a given state s via

$$h^*(s) = \min_{\sigma \in \trianglerighteq^*[\sigma(r|s[r])]} \left[cost(\sigma) + \sum_{v \in V \setminus \{r\}} \tilde{g}_{s[v];|\sigma|}(s[r]) \right] \tag{4.27}$$



r	$\|\sigma\|$	$cost(\sigma)$	$v{=}0$	$v{=}1$	$v{=}2$	$v{=}3$	$u{=}0$	$u{=}1$	$u{=}2$	$u{=}3$	$u{=}4$	$u{=}5$
0	1	0	100	∞	∞	0	201	200	101	100	1	0
	2	24	100	2	1	0	201	200	101	100	1	0
	3	48	100	2	1	0	53	102	3	2	1	0
	4	72	100	2	1	0	53	102	3	2	1	0
	5	96	3	2	1	0	5	4	3	2	1	0
	6	120	3	2	1	0	5	4	3	2	1	0
	7	144	3	2	1	0	5	4	3	2	1	0
1	1	0	∞	∞	1	0	∞	∞	∞	∞	∞	0
	2	24	100	∞	1	0	101	52	51	2	1	0
	3	48	3	2	1	0	101	52	51	2	1	0
	4	72	3	2	1	0	53	4	3	2	1	0
	5	96	3	2	1	0	53	4	3	2	1	0
	6	120	3	2	1	0	5	4	3	2	1	0
	7	144	3	2	1	0	5	4	3	2	1	0

(a) (b)

Figure 4.15: The database for a fork-structured problem with a binary-valued root variable r and two children v and u, and $G[r] = 0$, $G[v] = 3$, and $G[u] = 5$. (a) depicts the domain transition graphs of r (top), v (middle), and u (bottom); the numbers above and below each edge are the precondition on r and the cost of the respective action. (b) depicts the database created by the algorithm. For instance, the entry in row $r{=}0 \wedge |\sigma|{=}5$ and column $v = 0$ captures the value of $\tilde{g}_{v=0;5}(r{=}0)$ computed as in Eq. 4.26. The shaded entries are those examined during the online computation of $h^*(r{=}0, v{=}0, u{=}0)$.

where $\sigma(r|\vartheta_r)$ is defined similarly to $\sigma(r)$ but with respect to the initial value ϑ_r of r.

With the new formulation, the only computation that has to be performed online, per search node, is the final minimization over $\trianglerighteq^*[\sigma(r|s[r])]$ in Eq. 4.27, and this is the lightest part of the whole algorithm anyway. The major computations, notably those of $\{p_{\vartheta,\vartheta';\vartheta_r}\}$ and $\{\tilde{g}_{\vartheta;i}(\vartheta_r)\}$, can now be performed offline and shared between the evaluated states. The space required to store this information is $O(d^2|V|)$ as it contains only a fixed amount of information per pair of values of each variable. The time complexity of the offline computation is $O(d^3|V| + |A_r|)$; the $|A_r|$ component stems from precomputing the costs $cost(\sigma)$. The time complexity of the online computation per state is $O(d|V|)$; $|V|$ comes from the internal summation and d comes from the size of $\trianglerighteq^*[\sigma(r|s[r])]$. ■

Figure 4.15b shows the database created for a fork-structured problem with a binary-valued root r, two children v and u, and $G[r] = 0$, $G[v] = 3$, and $G[u] = 5$; the domain transition graphs of v and u are depicted in Figure 4.15(a). Online computation of $h^*(s)$ as in Eq. 4.27 for $s = (r{=}0, v{=}0, u{=}0)$ sums over the shaded entries of each of the four rows having such entries, and minimizes over the resulting four sums, with the minimum being obtained in the row $r{=}0 \wedge |\sigma|{=}5$.

4.5.2 Inverted Fork Databases

Theorem 12 *Let $\Pi = \langle V, A, I, G, cost\rangle$ be a planning task with an inverted fork causal graph with sink r and $|\mathcal{D}(r)| = b = O(1)$. There exists an h^*-partition for Π such that, for any set of k states, the time and space complexity of that h^*-partition is $O(b|V||A_r|^{b-1} + d^3|V| + k|V||A_r|^{b-1})$ and $O(|V||A_r|^{b-1} + d^2|V|)$, respectively, where $d = \max_v \mathcal{D}(v)$.*

Proof: Like the proof of Theorem 11, the proof of Theorem 12 is based on a modification of the polynomial-time algorithm for computing $h^*(s)$ used for the proof of Theorem 5 (Tractable Inverted Forks).

(1) For each parent variable $v \in V \setminus \{r\}$, and each pair of its values $\vartheta, \vartheta' \in \mathcal{D}(v)$, let $p_{\vartheta,\vartheta'}$ be the cost of the cheapest sequence of actions changing ϑ to ϑ'. The whole set $\{p_{\vartheta,\vartheta'}\}$ can be computed using the Floyd-Warshall algorithm on the domain transition graph of v in time $O(d^3|V|)$.

(2) Given a state s, for each cycle-free path $\pi = \langle a_1, \ldots, a_m \rangle$ from $s[r]$ to $G[r]$ in $DTG(v, \Pi)$, let g_π be the cost of the cheapest plan from s in Π based on π, and the shortest paths $\{p_{\vartheta,\vartheta'}\}$ computed in step (1). Each g_π can be computed as

$$g_\pi = \sum_{i=1}^{m} cost(a_i) + \sum_{i=0}^{m} \sum_{v \in V \setminus \{r\}} p_{\mathsf{pre}_i[v], \mathsf{pre}_{i+1}[v]},$$

where $\mathsf{pre}_0 \cdot \ldots \cdot \mathsf{pre}_{m+1}$ are the values required from the parents of r along the path π. That is, for each $v \in V \setminus \{r\}$, and $0 \leq i \leq m+1$,

$$\mathsf{pre}_i[v] = \begin{cases} s[v], & i = 0 \\ G[v], & i = m+1, \text{ and } G[v] \text{ is specified} \\ \mathsf{pre}(a_i)[v], & 1 \leq i \leq m, \text{ and } \mathsf{pre}(a_i)[v] \text{ is specified} \\ \mathsf{pre}_{i-1}[v] & \text{otherwise} \end{cases}.$$

From that, we have $h^*(s) = \min_\pi g_\pi$.

Note that step (1) is state-independent, but step (2) is not. However, the dependence of step (2) on the evaluated state can be substantially relaxed. As there are only $O(1)$ different values of r, it is possible to consider cycle-free paths to $G[r]$ from *all* values of r. For each such path π, and each parent variable $v \in V \setminus \{r\}$, we know what the first value of v required by π would be. Given that, we can precompute the cost-optimal plans induced by each π *assuming the parents start at their first required values.* The remainder of the computation of $h^*(s)$ is delegated to online, and the modified step (2) is as follows.

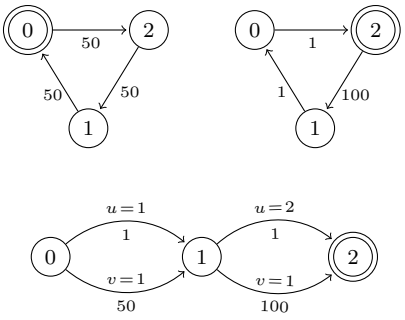
For each $\vartheta_r \in \mathcal{D}(r)$ and each cycle-free path $\pi = \langle a_1, \ldots, a_m \rangle$ from ϑ_r to $G[r]$ in $DTG(r, \Pi)$, let a "proxy" state s_π be

$$s_\pi[v] = \begin{cases} \vartheta_r, & v = r \\ G[v], & \forall 1 \leq i \leq m : \mathsf{pre}(a_i)[v] \text{ is unspecified} \\ \mathsf{pre}(a_i)[v], & i = \operatorname{argmin}_j \{\mathsf{pre}(a_j)[v] \text{ is specified}\} \end{cases},$$

that is, the nontrivial part of s_π captures the first values of $V \setminus \{r\}$ required along π.[7] Given that, let g_π be the cost of the cheapest plan from s_π in Π based on π, and the shortest paths $\{p_{\vartheta,\vartheta'}\}$ computed in (1). Each g_π can be computed as

$$g_\pi = \sum_{i=1}^{m} \left[w_{a_i} + \sum_{v \in V \setminus \{r\}} p_{\mathsf{pre}_i[v], \mathsf{pre}_{i+1}[v]} \right],$$

[7] For ease of presentation, we omit here the case where v is required neither along π, nor by the goal; such variables should be simply ignored when accounting for the cost of π.



(a)

r	π	s_π	g_π	$u=0$	$u=1$	$u=2$	$v=0$	$v=1$	$v=2$
0	$\langle a_1 \cdot a_3 \rangle$	$u=1, v=2$	202	100	0	50	1	2	0
	$\langle a_1 \cdot a_4 \rangle$	$u=1, v=1$	153	100	0	50	101	0	100
	$\langle a_2 \cdot a_3 \rangle$	$u=2, v=1$	153	50	100	0	101	0	100
	$\langle a_2 \cdot a_4 \rangle$	$u=0, v=1$	152	0	50	100	101	0	100
1	$\langle a_3 \rangle$	$u=2, v=2$	101	50	100	0	1	2	0
	$\langle a_4 \rangle$	$u=0, v=1$	102	0	50	100	101	0	100

(b)

Figure 4.16: The database for an inverted fork-structured problem with a $O(1)$ bounded sink variable r and two parents u and v, and $G[r]=2$, $G[u]=0$, and $G[v]=2$. (a) depicts the domain transition graphs of u (top left), v (top right), and r (bottom); the numbers above and below each edge are the preconditions and the cost of the respective action, respectively. (b) depicts the database created by the algorithm. The shaded entries are those examined during the online computation of $h^*(r=0, u=0, v=0)$.

where, for each $v \in V \setminus \{r\}$, and $1 \leq i \leq m+1$,

$$\mathsf{pre}_i[v] = \begin{cases} s_\pi[v], & i = 1 \\ G[v], & i = m+1, \text{ and } G[v] \text{ is specified} \\ \mathsf{pre}(a_i)[v], & 2 \leq i \leq m, \text{ and } \mathsf{pre}(a_i)[v] \text{ is specified} \\ \mathsf{pre}_{i-1}[v], & \text{otherwise} \end{cases}.$$

Storing the pairs (g_π, s_π) accomplishes the offline part of the computation. Now, given a search state s, we can compute

$$h^*(s) = \min_{\substack{\pi \text{ s.t.} \\ s_\pi[r]=s[r]}} \left[g_\pi + \sum_{v \in V \setminus \{r\}} p_{s[v], s_\pi[v]} \right]. \tag{4.28}$$

The number of cycle-free paths to $G[r]$ in $DTG(r, \Pi)$ is $\Theta(|A_r|^{b-1})$, and g_π for each such path π can be computed in time $O(b|V|)$. Hence, the overall offline time complexity is $O(b|V||A_r|^{b-1} + d^3|V|)$, and the space complexity (including the storage of the proxy states s_π) is $O(|V||A_r|^{b-1} + d^2|V|)$. The time complexity of the online computation per state via Eq. 4.28 is $O(|V||A_r|^{b-1})$; $|V|$ comes from the internal summation and $|A_r|^{b-1}$ from the upper bound on the number of cycle-free paths from $s[r]$ to $G[r]$. ∎

Figure 4.16(b) shows the database created for an inverted fork structured problem with a ternary-valued sink variable r, two parents u and v, and $G[r]=2$, $G[u]=0$, and $G[v]=2$. The domain transition graphs of u and v are depicted at the top of Figure 4.16(a); the actual identities of actions affecting these two parents are not important here. The actions affecting the sink r are

$$a_1 = \langle \{u=1, r=0\}, \{r=1\} \rangle$$
$$a_2 = \langle \{v=1, r=0\}, \{r=1\} \rangle$$
$$a_3 = \langle \{u=2, r=1\}, \{r=2\} \rangle$$
$$a_4 = \langle \{v=1, r=1\}, \{r=2\} \rangle.$$

domain	S	$h^{\mathcal{F}}$		$h^{\mathcal{J}}$		$h^{\mathcal{FJ}}$		MS-10^4		MS-10^5		HSP^*_{F}		Gamer		blind		$h_{\max}$	
		s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$
airport-ipc4	22	**22**	100	20	91	21	95	19	86	17	77	15	68	11	50	18	82	20	91
blocks-ipc2	30	21	70	18	60	18	60	18	60	20	67	**30**	100	**30**	100	18	60	18	60
depots-ipc3	7	**7**	100	4	57	**7**	100	**7**	100	4	57	4	57	4	57	4	57	4	57
driverlog-ipc3	12	**12**	100	**12**	100	**12**	100	**12**	100	**12**	100	9	75	11	92	7	58	8	67
freecell-ipc3	5	**5**	100	4	80	4	80	**5**	100	1	20	**5**	100	2	40	4	80	**5**	100
grid-ipc1	2	**2**	100	1	50	1	50	**2**	100	**2**	100	0	0	**2**	100	1	50	**2**	100
gripper-ipc1	20	7	35	7	35	7	35	7	35	7	35	6	30	**20**	100	7	35	7	35
logistics-ipc2	22	**22**	100	16	73	16	73	16	73	21	95	16	73	20	91	10	45	10	45
logistics-ipc1	7	**6**	86	4	57	5	71	4	57	5	71	3	43	**6**	86	2	29	2	29
miconic-strips-ipc2	85	51	60	50	59	50	59	54	64	55	65	45	53	**85**	100	50	59	50	59
mprime-ipc1	24	23	96	22	92	21	88	21	88	12	50	8	33	9	38	19	79	**24**	100
mystery-ipc1	21	**21**	100	18	86	**21**	100	17	81	13	62	11	52	8	38	18	86	18	86
openstacks-ipc5	7	**7**	100	**7**	100	**7**	100	**7**	100	**7**	100	**7**	100	**7**	100	**7**	100	**7**	100
pathways-ipc5	4	**4**	100	**4**	100	**4**	100	3	75	**4**	100	**4**	100	**4**	100	**4**	100	**4**	100
pipes-notank-ipc4	21	17	81	15	71	16	76	**20**	95	12	57	13	62	11	52	14	67	17	81
pipes-tank-ipc4	14	11	79	9	64	9	64	**13**	93	7	50	7	50	6	43	10	71	10	71
psr-small-ipc4	50	49	98	49	98	49	98	**50**	100	**50**	100	**50**	100	47	94	48	96	49	98
rovers-ipc5	7	6	86	**7**	100	6	86	6	86	**7**	100	6	86	5	71	5	71	6	86
satellite-ipc4	6	**6**	100	**6**	100	**6**	100	**6**	100	**6**	100	5	83	**6**	100	4	67	5	83
schedule-strips	46	**46**	100	40	87	**46**	100	22	48	1	2	11	24	3	7	29	63	31	67
tpp-ipc5	6	**6**	100	**6**	100	**6**	100	**6**	100	**6**	100	5	83	5	83	5	83	**6**	100
trucks-ipc5	9	6	67	7	78	7	78	6	67	5	56	**9**	100	3	33	5	56	7	78
zenotravel-ipc3	11	**11**	100	**11**	100	**11**	100	**11**	100	**11**	100	8	73	10	91	7	64	8	73
total	438	**368**		337		350		332		285		277		315		296		318	
$\hat{s}$		**20.56**		18.38		19.13		19.07		16.64		15.45		16.66		15.58		17.66	
w		**14**		7		9		11		9		6		8		2		6	

Table 4.7: A summary of the experimental results with databased versions of the fork-decomposition heuristics. Per domain, S denotes the number of tasks solved by *any* planner. Per planner/domain, the number of tasks solved by that planner is given both by the absolute number (s) and by the percentage from "solved by some planners" ($\%S$). Boldfaced results indicate the best performance within the corresponding domain. The last three rows summarize the number of solved instances, the domain-normalized measure of solved instances ($\hat{s}$), and the number of domains in which the planners achieved superior performance (w).

The domain transition graph of r is depicted at the bottom of Figure 4.16(a). Online computation of $h^*(s)$ as in Eq. 4.28 for $s = (r=0, v=0, u=0)$ sums over the shaded entries of each of the four rows having such entries, and minimizes over the resulting four sums, with the minimum being obtained in the lowest such row.

4.5.3 Experimental Evaluation

To evaluate the practical attractiveness of the databased fork-decomposition heuristics, we have repeated our empirical evaluation as in Section 4.4.2, but now for the databased versions of the heuristics. The detailed results of this evaluation are relegated to Tables B.7-B.12 in the appendix (pp. 183-188) but summarized here in Table 4.7. For each domain, the S column captures the number of tasks in that domain that were solved by at least one planner in the suite. Per planner/domain, the number of tasks solved by that planner is given both by the absolute number (s) and by the percentage from "solved by some planners" ($\%S$). Boldfaced results indicate the best performance within the corresponding domain. The last three rows summarize the performance of the planners via three measures. The first is the number of tasks solved in all the 23 domains; this is basically the performance evaluation measure used in the optimization track at IPC-2008. As domains are not equally challenging and do not equally discriminate between the planners' performance, the second is a "domain-normalized" performance measure

$$\hat{s}(p) = \sum_{\text{domain } \mathcal{D}} \frac{\#\text{tasks in } \mathcal{D} \text{ solved by planner } p}{\#\text{tasks in } \mathcal{D} \text{ solved by some planners}}.$$

domain	S	$h^{\mathcal{F}}$		$h^{\mathcal{J}}$		$h^{\mathcal{FJ}}$		HSP^*_{F}		Gamer		blind	
		s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$	s	$\%S$
elevators-strips-ipc6	22	18	82	14	64	15	68	7	32	**22**	100	11	50
openstacks-strips-ipc6	21	19	90	19	90	19	90	**21**	100	19	90	19	90
parcprinter-strips-ipc6	16	14	88	13	81	13	81	**16**	100	9	56	10	63
pegsol-strips-ipc6	27	**27**	100	**27**	100	**27**	100	**27**	100	24	89	**27**	100
scanalyzer-strips-ipc6	12	**12**	100	6	50	6	50	6	50	11	92	**12**	100
sokoban-strips-ipc6	28	25	89	26	93	**27**	96	13	46	20	71	20	71
transport-strips-ipc6	11	**11**	100	**11**	100	**11**	100	9	82	**11**	100	**11**	100
woodworking-strips-ipc6	14	8	57	8	57	8	57	9	64	**14**	100	7	50
total	152	**134**		124		126		108		130		117	
$\hat{s}$		**7.06**		6.35		6.43		5.74		6.99		6.24	
w		**3**		2		**3**		**3**		**3**		**3**	

Table 4.8: A summary of the experimental results. Per domain, S denotes the number of tasks solved by *any* planner. Per planner/domain, the number of tasks solved by that planner is given both by the absolute number (s) and by the percentage from "solved by some planners" ($\%S$). Boldfaced results indicate the best performance within the corresponding domain. The last three rows summarize the number of solved instances, the domain-normalized measure of solved instances ($\hat{s}$), and the number of domains in which the planners achieved superior performance (w).

Finally, the third measure corresponds to the number of domains w in which the planner in question solved at least as many tasks as any other planner.

Overall, Table 4.7 clearly suggests that heuristic search with "databased" fork-decomposition heuristics favorably competes with the state of the art of optimal planning. In particular, A^* with the "only forks" heuristic $h^{\mathcal{F}}$ exhibited the best overall performance according to all three measures. In terms of the absolute number of solved instances, A^* with all three fork-decomposition heuristics outperformed all other planners in the suite. The contribution of databasing to the success of the fork-decomposition heuristics was dramatic. Looking back at the results with "fully online" heuristic computation depicted in Table 4.6, note that the total number of solved instances for the fork-decomposition heuristics $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$ increased by 74, 55, and 76, respectively, and this made the whole difference.

We have also performed a comparative evaluation on the planning domains from the recent IPC-2008. The IPC-2008 domains differ from the previous domains in that actions had various costs, and, more importantly, many actions had zero cost. The latter is an issue for heuristic-search planners because heuristic functions cannot differentiate between subplans that have the same cost of zero, but differ in length. In any case, the comparative side of our evaluation on the IPC-2008 domains differ on several points from the previous one. First, neither for merge-and-shrink nor for $h_{\max}$ heuristics, we had implementation supporting arbitrary action costs. Hence, our comparison here is only with Gamer, HSP^*_{F}, and blind search. Second, to ensure admissibility of the blind search, the latter has been modified to return on non-goal states the cost of the cheapest applicable action. Finally, all the planners were run on a 3GHz Intel E8400 CPU with 4 GB memory, using 2 GB memory limit and 30 minute timeout. The results of this evaluation are summarized in Table 4.8; for the detailed results we refer the reader to Tables B.13-B.14 in the appendix (pp. 189-190). Overall, these results show that A^* with the fork-decomposition heuristics are very much competitive on the IPC-2008 domains as well.

Chapter 5

Abstraction-based Heuristics Composition

Since the late 1990s, numerous admissible heuristics for domain-independent planning have been proposed and found practically effective, with research in this direction continuously expanding. Of course, as planning is known to be NP-hard even for extremely conservative planning formalisms (Bylander, 1994), no heuristic should be expected to work well in all planning tasks. Moreover, even for a fixed planning task, no tractable heuristic will home in on all the "combinatorics" of the task at hand. The promise, however, is that different heuristics will target different sources of the planning complexity, and composing a set of heuristics to exploit their individual strengths could allow a larger range of planning tasks to be solved and each individual task to be solved more efficiently.

In this chapter we focus on the fundamental question of how one should better compose a set of admissible heuristics in solving a given planning task. One of the well-known and heavily-used properties of admissible heuristics is that taking the maximum of their values maximizes informativeness while preserving admissibility. A more recent, alternative approach to composing a set of admissible heuristics corresponds to carefully separating the information used by the different heuristics in the set so that their values could be summed instead of maximized over. This direction was first exploited in the works on additive pattern database (PDB) heuristics (Edelkamp, 2001; Felner et al., 2004; Haslum et al., 2007), and more recently, in the scope of constrained PDBs, m-reachability, and implicit abstraction (also known as structural-pattern) heuristics (Haslum et al., 2005; Katz & Domshlak, 2008c).

The basic idea underlying all these *additive heuristic ensembles* is elegantly simple: for each planning task's action a, if it can possibly be counted by more than one heuristic in the ensemble, then one should ensure that the cumulative counting of the cost of a does not exceed its true cost in the original task. Such *action cost partitioning* was originally achieved by accounting for the whole cost of each action in computing a single heuristic in the ensemble, while ignoring the cost of that action in computing all the other heuristics in the ensemble (Edelkamp, 2001; Felner et al., 2004; Haslum et al., 2005). Recently, this "all-in-one/nothing-in-rest" action-cost partitioning has been generalized by Katz and Domshlak (2007a, 2008c) and Yang et al. (2007, 2008) to *arbitrary* partitioning of the action cost among the heuristics in the ensemble.

The great flexibility of additive heuristic ensembles, however, is a mixed blessing. For better or for worse, the methodology of taking the maximum over the values provided by an arbitrary set of

independently constructed admissible heuristics is entirely nonparametric. By contrast, switching to additive heuristic ensembles requires *selecting an action-cost partitioning scheme*, and this decision problem poses a number of computational challenges:

- The space of alternative action-cost partitions is infinite as the cost of each action can be partitioned into an arbitrary set of nonnegative real numbers, the sum of which does not exceed the cost of that action.
- At least in domain-independent planning, our goal is a fully unsupervised decision process.
- Last but not least, the relative quality of each action-cost partition (in terms of the accuracy of the resulting additive heuristic) may vary dramatically between the examined search states. Hence, the choice of the action-cost partitioning scheme should ultimately be a function of the search state in question.

These concerns may explain why all previous works on both domain-specific and domain-independent additive heuristic ensembles adopt this or another ad hoc, *fixed* choice of action-cost partition. Consequently, all the reported empirical comparative evaluations of various max-based and additive heuristic ensembles are inconclusive—for some search states along the search process, the (pre-selected) additive heuristic was found to dominate the max-combination, while for the other states the opposite was the case. In the context of domain-specific additive PDBs, Yang et al. (2007) conclude that "determining which abstractions [here: action-cost partitioning schemes] will produce additives that are better than max over standards is still a big research issue."

Focusing on abstraction heuristics, our contribution in this chapter is precisely in addressing the problem of choosing the right action-cost partitioning over a given set of heuristics:

1. We provide a procedure that, given (i) a classical planning task Π, (ii) a forward-search state s of Π, and (iii) a set of heuristics based on abstractions of Π, derives an *optimal action-cost partition for* s, that is, a partition that maximizes the heuristic estimate of that state. The procedure is *fully unsupervised*, and is based on a linear programming formulation of that optimization problem.

2. We show that the time complexity of our procedure is *polynomial* for arbitrary sets of *all* abstraction-based heuristic functions with which we are familiar. Such "procedure-friendly" heuristics include PDBs (Edelkamp, 2001; Yang et al., 2007), constrained PDBs (Haslum et al., 2005), merge-and-shrink abstractions (Helmert et al., 2007), fork-decomposition implicit abstractions (Katz & Domshlak, 2008c), and implicit abstractions based on tractable constraint optimization over tree-shaped constraint networks (Katz & Domshlak, 2008a). Note that the estimate provided by a max-based ensemble corresponds to the estimate provided by the respective additive ensemble under *some* action-cost partitioning. Thus, by finding an *optimal* action-cost partition we provide a formally complete answer to the aforementioned question of "to add or not to add" in the context of abstraction heuristics.

Taking the fork-decomposition abstractions as a case study, we evaluate the empirical effectiveness of switching from ad hoc to optimal additive composition. Our evaluation on a wide range of International Planning Competition (IPC) benchmarks shows a substantial reduction in the number of nodes expanded by the A^* algorithm. However, in the standard time-bounded setting, this reduction in expanded nodes is typically negatively balanced by the much more expensive

per-node computation of the optimal additive heuristic. To overcome this pitfall without forfeiting the promise of optimized action cost partitions altogether, we suggest that optimal action cost partitions be derived only with respect to a subset of evaluated nodes. We examine this approach empirically in its extreme setting where only one optimal action cost partition is computed per planning task: the one that is optimal for the task's initial state. This action cost partition is then used for all the states evaluated by the A^* algorithm. Our experiments show that even such a conservative use of optimization results in substantial improvement over the same heuristic-search planner relying on an ad hoc action cost partition.

Going back to additive abstractions, Definition 3 on page 12 poses only a general requirement of not overestimating the path costs between the states. For action cost partitions, we need to more tightly bind the original and abstract TG-structures by

(i) associating each abstract transition label with a single original transition label, and

(ii) verifying that, in each individual abstraction, each original transition is represented by an abstract transition *path* of a certain form.

These structural requirements are captured by the notion of *ABS-ensemble* defined below.

Definition 16 *An* **ensemble of abstractions** *(***ABS-ensemble***) of a TG-structure* $\mathcal{T} = (S, L, Tr, s^I, S^G)$ *is a set of triplets* $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^{k}$ *where, for* $1 \leq i \leq k$*,*

- $\mathcal{T}_i = (S_i, L_i, Tr_i, s_i^I, S_i^G)$ *is a TG-structure,*
- $\alpha_i : S \to S_i$ *is an abstraction mapping with* $\alpha_i(s^I) = s_i^I$*, and* $\alpha_i(s) \in S_i^G$ *for all* $s \in S^G$*,*
- $\beta_i : L_i \to L$ *is an* action association mapping*, and*
- *for each transition* $\langle s, l, s' \rangle \in Tr$*, there is a path* ρ_i *from* $\alpha_i(s)$ *to* $\alpha_i(s')$ *in* $\mathcal{T}_i$ *such that*

  (i) all the transitions on ρ_i *have different labels, and*

  (ii) for all labels l' *along* ρ_i*, it holds that* $\beta_i(l') = l$*.*

The notion of ABS-ensembles generalizes the qualitative skeletons of various known additive abstractions that are based on action-cost partitioning, as follows:

(1) The setting of all β_i being bijective mappings captures additive *homomorphic abstractions* such as standard and constrained pattern database (Yang et al., 2008; Haslum et al., 2005), and merge-and-shrink abstractions (Helmert et al., 2007).

(2) The setting of all α_i being injective (with, possibly, $|S_i| > |S|$) captures additive *embedding abstractions*, obtained by expanding the original action set by some new actions derived from the original ones. In such cases, the new actions are constructed with certain desired properties such as positive and/or unary effects only (Bylander, 1994).

(3) Each individual abstraction in an ABS-ensemble may correspond to a *hybrid homomorphic/embedding abstraction* such as those induced by some implicit abstractions (Katz & Domshlak, 2008c).

Of course, nothing in the definition of ABS-ensemble prevents us from using an *arbitrary mixture* of the above three types of abstractions. First things first, however. Theorem 13 connects ABS-ensembles to the additive abstractions induced by them via action-cost partitioning. It is worth noting here that the generality of Definition 16 and Theorem 13 is not an exercise only—later we exploit it to establish some computational results of an adequate generality.

Theorem 13 (Additive Abstractions) *Let $\mathcal{T}$ be a TG-structure with labels L, and let $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^k$ be an ABS-ensemble of $\mathcal{T}$. For any function $\varpi : L \to \mathbb{R}^{0+}$, and any set of functions $\varpi_i : L_i \to \mathbb{R}^{0+}$, $1 \leq i \leq k$, such that*

$$\forall l \in L : \sum_{i=1}^{k} \sum_{l' \in \beta_i^{-1}(l)} \varpi_i(l') \leq \varpi(l), \tag{5.1}$$

$\mathcal{A} = \{\langle \langle \mathcal{T}_i, \varpi_i \rangle, \alpha_i \rangle\}_{i=1}^k$ is an additive abstraction of the transition graph $\langle \mathcal{T}, \varpi \rangle$.

Proof: Let $\mathcal{T} = (S, L, Tr, s^I, S^G)$ be a TG-structure, $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^k$ be an ABS-ensemble of $\mathcal{T}$, and $\varpi : L \to \mathbb{R}^{0+}$, $\varpi_i : L_i \to \mathbb{R}^{0+}$, $1 \leq i \leq k$ be some functions satisfying Eq. 5.1. To prove that $\mathcal{A} = \{\langle \langle \mathcal{T}_i, \varpi_i \rangle, \alpha_i \rangle\}_{i=1}^k$ is an additive abstraction of the transition graph $\langle \mathcal{T}, \varpi \rangle$ we should show that Eq. 2.1 holds for each pair of states $s, s' \in S$.

Let $\rho = \langle s_0, l_1, s_1 \rangle, \langle s_1, l_2, s_2 \rangle, \ldots, \langle s_{m-1}, l_m, s_m \rangle$ be a cheapest path from $s = s_0$ to $s' = s_m$ in $\langle \mathcal{T}, \varpi \rangle$. Thus, $cost(s, s') = \sum_{j=1}^m \varpi(l_j)$. For each $1 \leq i \leq k$ and $1 \leq j \leq m$, let $\rho_{i,j}$ be a simple, label-disjoint path from $\alpha_i(s_{j-1})$ to $\alpha_i(s_j)$ in $\mathcal{T}_i$ along transitions labeled only with labels in $\beta_i^{-1}(l_j)$; the existence of such a $\rho_{i,j}$ is guaranteed by Definition 16. Thus, for $1 \leq i \leq k$, the concatenation $\rho_i = \rho_{i,1} \cdot \ldots \cdot \rho_{i,m}$ is a (not necessarily cheapest) transition path from $\alpha_i(s)$ to $\alpha_i(s')$ in $\mathcal{T}_i$.

Given that, we have

$$cost(\alpha_i(s), \alpha_i(s')) \leq \sum_{l' \in \rho_i} \varpi_i(l') = \sum_{j=1}^{m} \sum_{l' \in \rho_{i,j}} \varpi_i(l') \leq \sum_{j=1}^{m} \sum_{l' \in \beta_i^{-1}(l_j)} \varpi_i(l')$$

and, summing over all $1 \leq i \leq k$, we have

$$\sum_{i=1}^{k} cost(\alpha_i(s), \alpha_i(s')) \leq \sum_{i=1}^{k} \sum_{j=1}^{m} \sum_{l' \in \beta_i^{-1}(l_j)} \varpi_i(l') = \sum_{j=1}^{m} \sum_{i=1}^{k} \sum_{l' \in \beta_i^{-1}(l_j)} \varpi_i(l') \overset{(*)}{\leq} \sum_{j=1}^{m} \varpi(l_j) = cost(s, s'),$$

where inequality $(*)$ follows from Eq. 5.1 and label-disjointness stated by condition *(i)* in Definition 16. ∎

Definition 17 *Let $\Pi = \langle V, A, I, G, cost \rangle$ be a planning task, $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^k$ be an ABS-ensemble of $\mathcal{T}(\Pi)$, and $\{\varpi_i : L_i \to \mathbb{R}^{0+}\}_{i=1}^k$ be a set of transition cost functions. Then $\mathcal{A} = \{\langle \langle \mathcal{T}_i, \varpi_i \rangle, \alpha_i \rangle\}_{i=1}^k$ is an* **additive abstraction of Π with respect to $\mathcal{AE}$**, *denoted by $\mathcal{A} \in_\Pi \mathcal{AE}$, if*

$$\forall a \in A : \sum_{i=1}^{k} \sum_{l \in \beta_i^{-1}(a)} \varpi_i(l) \leq cost(a). \tag{5.2}$$

In other words, each additive abstraction $\mathcal{A} \in_\Pi \mathcal{AE}$ corresponds to a certain action-cost partitioning of Π over $\mathcal{AE}$ and, importantly, vice versa.

Definition 18 *Let Π be a planning task with state set S, and let $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^k$ be an ABS-ensemble of $\mathcal{T}(\Pi)$.*

- *For any additive abstraction $\mathcal{A} = \{\langle \langle \mathcal{T}_i, \varpi_i \rangle, \alpha_i \rangle\}_{i=1}^k \in_\Pi \mathcal{AE}$, the* **additive heuristic** *$h_\mathcal{A}$ is the function assigning to each state $s \in S$ the quantity $\sum_{i=1}^k cost(\alpha_i(s), S_i^G)$.*
- *The* **optimal additive heuristic** *$h_\mathcal{AE}$ is the function assigning to each state $s \in S$ the quantity $\max_{\mathcal{A} \in_\Pi \mathcal{AE}} h_\mathcal{A}(s)$.*

Definition 18 specifies the set of *all* additive heuristics for Π obtainable via action-cost partitioning over a given ABS-ensemble, as well as a *tight upper bound* $h_\mathcal{AE}(s)$ on the heuristic estimate obtainable for a state s from that infinite set of additive heuristics. The admissibility of each heuristic $h_\mathcal{A}$ in that space is immediate from Definition 3, and thus the proof of Theorem 14 below is straightforward.

Theorem 14 (Admissibility of $h_\mathcal{AE}$) *For any planning task Π, any ABS-ensemble $\mathcal{AE}$ of $\mathcal{T}(\Pi)$, and any state s of Π, we have $h_\mathcal{AE}(s) \leq h^*(s)$.*

Proof: Let $\Pi = \langle V, A, I, G, cost \rangle$ be a planning task, $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^k$ be an ABS-ensemble of $\mathcal{T}(\Pi)$, and s be some state of Π. From Definitions 3 and 18, for any additive abstraction $\mathcal{A} \in_\Pi \mathcal{AE}$, we have

$$h_\mathcal{A}(s) = \sum_{i=1}^k cost(\alpha_i(s), S_i^G) \leq cost(s, S^G) = h^*(s),$$

which holds, in particular, for $\mathcal{A}$ which maximizes $h_\mathcal{A}(s)$ over all $\mathcal{A}' \in_\Pi \mathcal{AE}$, that is, for $h_\mathcal{AE}(s)$. ■

5.1 LP-Optimizable Ensembles of Abstractions

Having specified the notion of optimal additive heuristic $h_\mathcal{AE}$, we now proceed with the computational side of the story. Suppose we are given a planning task $\Pi = \langle V, A, I, G, cost \rangle$ over states S, and an ABS-ensemble $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^k$ of $\mathcal{T}(\Pi)$. Assuming that the individual additive heuristics $h_\mathcal{A}$ corresponding to arbitrary fixed action cost partitions can be efficiently computed for all $\mathcal{A} \in_\Pi \mathcal{AE}$ (as it is the case with the ABS-ensembles of practical interest), the main question is whether $h_\mathcal{AE}$ can be efficiently computed as well. At first glance, the infinite space of alternative action cost partitions, as well as the search-state dependence of their relative quality, do not offer much reason for optimism. Here, however, we characterize a family of ABS-ensembles for which the answer to that question is actually affirmative. This characterization is constructive in terms of *computability of $h_\mathcal{AE}(s)$ via a compact linear program induced by the triplet of a planning task Π, an ABS-ensemble $\mathcal{AE}$ of $\mathcal{T}(\Pi)$, and a search state s of Π*. For the sake of readability, and with the exception of formal claims, we use "compact" as a synonym for "being of size $O(poly(||\Pi||))$".

We begin with introducing a set of linear constraints specifying all possible cost partitions of the actions $a \in A$ over their representatives $\bigcup_{i=1}^k \beta_i^{-1}(a)$ in the components of $\mathcal{AE}$. For each abstract label $l \in \bigcup_{i=1}^k L_i$, let w_l be a nonnegative real-valued variable uniquely associated with l, and let the set of all these "label-cost" variables be denoted by $\overrightarrow{w}$. The (linear) **additivity constraint $\mathbb{C}^{\mathsf{add}}(\overrightarrow{w})$ of Π on $\mathcal{AE}$** mirrors Eq. 5.2 as

$$\forall a \in A: \quad \sum_{i=1}^k \sum_{l \in \beta_i^{-1}(a)} w_l \leq cost(a). \tag{5.3}$$

We denote by $\mathscr{H}^{\mathsf{add}}$ the convex polyhedron specified by $\mathbb{C}^{\mathsf{add}}(\overrightarrow{w})$. Note that there is a straightforward bijective correspondence between the points $\mathbf{w} \in \mathscr{H}^{\mathsf{add}}$ and (with some abuse of notation) the additive abstractions $\mathcal{A}_{\mathbf{w}} = \{\langle\langle \mathcal{T}_i, \mathbf{w}\rangle, \alpha_i\rangle\}_{i=1}^{k} \in_{\Pi} \mathcal{AE}$. Using the additivity constraint $\mathbb{C}^{\mathsf{add}}(\overrightarrow{w})$ as a building block, we now proceed with characterizing our "$h_{\mathcal{AE}}$-friendly" family of *LP-optimizable* ABS-ensembles.

Definition 19 *Let Π be a planning task, $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i\rangle\}_{i=1}^{k}$ be an ABS-ensemble of $\mathcal{T}(\Pi)$, and $\mathbb{C}^{\mathsf{add}}(\overrightarrow{w})$ be the additivity constraint of Π on $\mathcal{AE}$.*

1. *Given a state s of Π, an* **LP-encoding of $\mathcal{AE}$ with respect to** *s is a triplet $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw})\rangle$ where $\overrightarrow{x}$ is a set of nonnegative real-valued variables, f is a real-valued affine function over $\overrightarrow{x}$, $\mathbb{C}^{\mathcal{AE}}$ is a set of linear constraints on $\overrightarrow{x}$ and $\overrightarrow{w}$, and*

$$\forall \mathbf{w} \in \mathscr{H}^{\mathsf{add}} : \quad \max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} f(\mathbf{x}) \quad = h_{\mathcal{A}_{\mathbf{w}}}(s), \tag{5.4}$$

   where $\mathscr{H}^{\mathcal{AE}}$ is the convex polyhedron specified by $\mathbb{C}^{\mathcal{AE}} \cup \mathbb{C}^{\mathsf{add}}$.

2. *The ABS-ensemble $\mathcal{AE}$ is called* **LP-optimizable** *if, for every state s of Π, there exists (and one can generate in poly-time) a compact LP-encoding $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw})\rangle$ of $\mathcal{AE}$ with respect to s.*

The next two theorems provide two important ingredients for characterizing our "$h_{\mathcal{AE}}$-friendly" family of ABS-ensembles on the grounds of their LP-optimizability.

Theorem 15 (Tractability of $h_{\mathcal{AE}}$) *For any planning task Π, and any LP-optimizable ABS-ensemble $\mathcal{AE}$ of $\mathcal{T}(\Pi)$, the optimal additive heuristic $h_{\mathcal{AE}}(s)$ is poly-time computable for every state s of Π.*

Proof: Let $\Pi = \langle V, A, I, G, cost\rangle$ be a planning task, $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i\rangle\}_{i=1}^{k}$ be an LP-optimizable ABS-ensemble of $\mathcal{T}(\Pi)$, and s be a state of Π. Let $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw})\rangle$ be an LP-encoding of $\mathcal{AE}$ with respect to s. Consider now the linear program $\mathcal{L}$ defined by the variables $\overrightarrow{xw}$, constraints $\mathbb{C}^{\mathcal{AE}} \cup \mathbb{C}^{\mathsf{add}}$, and the maximized objective function f. If $\mathbf{xw}$ is a solution for $\mathcal{L}$, then

$$f(\mathbf{x}) = \max_{\mathbf{w} \in \mathscr{H}^{\mathsf{add}}} \max_{\mathbf{x'w} \in \mathscr{H}^{\mathcal{AE}}} f(\mathbf{x'}) \overset{\text{Eq.5.4}}{=} \max_{\mathbf{w} \in \mathscr{H}^{\mathsf{add}}} h_{\mathcal{A}_{\mathbf{w}}}(s) = \max_{\mathcal{A} \in_{\Pi} \mathcal{AE}} h_{\mathcal{A}}(s) = h_{\mathcal{AE}}(s). \tag{5.5}$$

Moreover, by Definition 19, both the number of variables and constraints in $\mathcal{L}$ is $O(poly(||\Pi||))$. Thus, the claim follows from Eq. 5.5 and poly-time solvability[1] of linear programming (Schrijver, 1998). ∎

Theorem 16 (Composition) *For any planning task Π, and any set of LP-optimizable ABS-ensembles $\{\mathcal{AE}_1, \ldots, \mathcal{AE}_m\}$ of $\mathcal{T}(\Pi)$, if $m = O(poly(||\Pi||))$, then the joint ABS-ensemble $\mathcal{AE} = \bigcup_{i=1}^{m} \mathcal{AE}_i$ of $\mathcal{T}(\Pi)$ is also LP-optimizable.*

[1]The notion of LP-optimizability can possibly be further generalized by requiring only that the linear program induced by $\mathbb{C}^{\mathcal{AE}} \cup \mathbb{C}^{\mathsf{add}}$ and f be solvable in time $O(poly(||\Pi||))$. For instance, this will allow capturing poly-time solvable linear programs with an exponential number of constraints but a poly-time separation oracle. However, it is unclear to us at this stage whether this generalization will fit the important claim of "composition" in Theorem 16.

Proof: Let Π and $\{\mathcal{AE}_1, \ldots, \mathcal{AE}_m\}$ be as in the claim, and let s be some state of Π. For $1 \leq j \leq m$, let $\mathbb{L}_j(s) = \langle \overrightarrow{x_j}, f_j(\overrightarrow{x_j}), \mathbb{C}_j^{\mathcal{AE}}(\overrightarrow{x_j w_j}) \rangle$ be a compact LP-encoding of the ABS-ensemble $\mathcal{AE}_j$.

First, we specify a composite additivity constraint $\mathbb{C}^{\mathsf{add}}(\overrightarrow{w})$ over $\overrightarrow{w} = \bigcup_{j=1}^{m} \overrightarrow{w_j}$ as

$$\forall a \in A: \quad \sum_{j=1}^{m} \sum_{i=1}^{|\mathcal{AE}_j|} \sum_{l \in \beta_{j,i}^{-1}(a)} w_l \quad \leq \quad cost(a).$$

Given that, the LP-encoding $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw}) \rangle$ of $\mathcal{AE} = \bigcup_{j=1}^{m} \mathcal{AE}_j$ is set to

- $\overrightarrow{x} = \bigcup_{j=1}^{m} \overrightarrow{x_j}$,
- $f(\overrightarrow{x}) = \sum_{j=1}^{m} f_j(\overrightarrow{x_j})$, and
- $\mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw}) = \bigcup_{j=1}^{m} \mathbb{C}_j^{\mathcal{AE}}(\overrightarrow{x_j w_j})$.

For any point $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_m) \in \mathscr{H}^{\mathsf{add}}$, and any assignment $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ to $\overrightarrow{x}$, if $\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}$, then we have $\mathbf{x}_j \mathbf{w}_j \in \mathscr{H}^{\mathcal{AE}_j}$ for all $1 \leq j \leq m$. Therefore, we have

$$\forall \mathbf{w} \in \mathscr{H}^{\mathsf{add}}: \quad \max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} f(\mathbf{x}) = \max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} \sum_{j=1}^{m} f_j(\mathbf{x}_j) \overset{(*)}{=} \sum_{j=1}^{m} \max_{\mathbf{x}'_j \mathbf{w}_j \in \mathscr{H}^{\mathcal{AE}_j}} f_j(\mathbf{x}'_j) = \sum_{j=1}^{m} h_{\mathcal{A}_{\mathbf{w}_j}}(s) = h_{\mathcal{A}_{\mathbf{w}}}(s),$$

where the equality $(*)$ stems from that, having fixed the value of variables $\overrightarrow{w}$, optimizations of the individual objective functions $f_1, \ldots, f_m$ are pair-wise independent due to the pair-wise disjointness of the variable sets $\overrightarrow{x_1}, \ldots, \overrightarrow{x_m}$. In other words, for each point $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_m) \in \mathscr{H}^{\mathsf{add}}$ on the convex polyhedron specified by $\mathbb{C}^{\mathsf{add}}(\overrightarrow{w})$, we have $\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}$ if and *only if* $\mathbf{x}_j \mathbf{w}_j \in \mathscr{H}_j^{\mathcal{AE}}$ for all $1 \leq j \leq m$. Finally, it is immediate that both $|\overrightarrow{xw}|$ and $|\mathbb{C}^{\mathcal{AE}} \cup \mathbb{C}^{\mathsf{add}}|$ are $O(poly(||\Pi||))$, and thus $\mathcal{AE}$ is LP-optimizable. ∎

While the two key properties of LP-optimizable ABS-ensembles stated by Theorems 15 and 16 are gratifying, having read this far the reader may rightfully ask whether any of the known families of abstraction heuristics actually lead to LP-optimizable ABS-ensembles. The answer to this question provided in what follows turns out to be positive to a surprising extent.

5.2 LP-Optimization and Explicit Abstractions

As we already mentioned in Section 2.2.3, the most well-studied abstraction heuristics correspond to explicit abstractions, characterized by relatively small, and thus explicitly searchable, abstract spaces. This class of abstractions contains different variants of pattern databases (Culberson & Schaeffer, 1998; Felner et al., 2004; Edelkamp, 2001; Haslum et al., 2005, 2007), variable-domain abstractions (Hernadvölgyi & Holte, 1999; Domshlak et al., 2009), and merge-and-shrink abstractions (Helmert et al., 2007).

Definition 20 *An ABS-ensemble $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^{k}$ of $\mathcal{T}(\Pi)$ for a planning task Π is an* **explicit ABS-ensemble** *if $k = O(poly(||\Pi||))$ and, for $1 \leq i \leq k$, $||\mathcal{T}_i|| = O(poly(||\Pi||))$.*

For any explicit ABS-ensemble $\mathcal{AE}$, any explicit abstraction $\mathcal{A} = \{\langle\langle\mathcal{T}_i, \varpi_i\rangle, \alpha_i\rangle\}_{i=1}^k \in_\Pi \mathcal{AE}$ induces an admissible additive heuristic $h_\mathcal{A}$. So far, however, how to choose this abstraction, that is, the actual choice of the action-cost partitioning, has remained an open issue, and the choices were made on an ad hoc basis (Yang et al., 2007). This is exactly where LP-optimization comes into the picture.

Computing $h_\mathcal{A} = \sum_{i=1}^k cost(\alpha_i(s), S_i^G)$ for a *fixed* explicit abstraction $\mathcal{A} = \{\langle\langle\mathcal{T}_i, \varpi_i\rangle, \alpha_i\rangle\}_{i=1}^k \in_\Pi \mathcal{AE}$ is usually done by computing each $cost(\alpha_i(s), S_i^G)$ using either breadth-first search or the Dijkstra algorithm over the *explicitly constructed* transition graph $\langle\mathcal{T}_i, \varpi_i\rangle$. However, the corresponding single-source shortest paths (SSSP) problem also has an elegant LP formulation. Given a directed graph $\mathcal{G} = (N, E)$ and a source node $v \in N$, if $d(v')$ is a variable corresponding to the shortest-path length from v to v', then the solution of the linear program

$$\begin{aligned} &\max_{\overrightarrow{d}} \sum_{v'} d(v') \\ &\text{s.t. } d(v) = 0 \\ &\qquad d(v'') \leq d(v') + w(v', v''), \ \ \forall (v', v'') \in E \end{aligned} \tag{5.6}$$

induces a solution to the SSSP problem over $\mathcal{G}$ and source v.

The LP formulation of the SSSP problem is not widely used as the aforementioned graph-search techniques resolve SSSP much more efficiently. However, this LP formulation is precisely what we need to bridge the gap between explicit abstractions and optimal action cost partition. A compact LP-encoding of an explicit ABS-ensemble $\mathcal{AE} = \{\langle\mathcal{T}_i, \alpha_i, \beta_i\rangle\}_{i=1}^k$ can now be obtained by

(1) putting together the linear constraints as in Eq. 5.6 for TG-structures $\mathcal{T}_i$,

(2) replacing the edge-weight constants $w(v', v'')$ by variables associated with the corresponding transition labels, and

(3) constraining the latter label-cost variables with the proper additivity constraints.

Given a planning task Π, an explicit ABS-ensemble $\mathcal{AE} = \{\langle\mathcal{T}_i, \alpha_i, \beta_i\rangle\}_{i=1}^k$ of $\mathcal{T}(\Pi)$, and a state s of Π, the LP-encoding $\mathbb{L} = \langle\overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw})\rangle$ of $\mathcal{AE}$ with respect to s is constructed as follows. First, let the label-cost variables $\overrightarrow{w}$ contain a variable w_l for every abstract label $l \in \bigcup_{i=1}^k L_i$. The additivity constraint is defined in terms of these label-cost variables $\overrightarrow{w}$ exactly as in Eq. 5.3. The components of the LP-encoding $\mathbb{L}$ are then set to

$$\begin{aligned} \overrightarrow{x} &= \bigcup_{i=1}^k \{d(\sigma) \mid \sigma \in S_i\} \cup \{d(G_i)\} \\ \mathbb{C}^{\mathcal{AE}} &= \begin{cases} d(\sigma) \leq d(\sigma') + w_l, & \forall\langle\sigma', l, \sigma\rangle \in Tr_i \\ d(\sigma) = 0, & \sigma = \alpha_i(s) \\ d(G_i) \leq d(\sigma), & \sigma \in S_i^G \end{cases} , \forall i \\ f(\overrightarrow{x}) &= \sum_{i=1}^k d(G_i). \end{aligned} \tag{5.7}$$

Since each TG-structure $\mathcal{T}_i$ in explicit ABS-ensemble $\mathcal{AE}$ is of size $O(poly(||\Pi||))$, it is immediate that the LP-encoding $\mathbb{L} = \langle\overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw})\rangle$ is both compact and poly-time constructible for any state s of Π, giving us all we need to prove Theorem 17 below.

Theorem 17 *Optimal additive heuristic $h_{\mathcal{AE}}(s)$ is poly-time computable for any planning task Π, any explicit ABS-ensemble $\mathcal{AE}$, and any state s of Π.*

Proof: Let Π be a planning task, $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^k$ be an explicit ABS-ensemble of $\mathcal{T}(\Pi)$, s be some state of Π, $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw}) \rangle$ be as in Eq. 5.7, and $\mathbf{w}$ be some point in $\mathscr{H}^{\mathsf{add}}$. Given that, we have $\mathcal{A}_{\mathbf{w}} = \{\langle \langle \mathcal{T}_i, \mathbf{w} \rangle, \alpha_i \rangle\}_{i=1}^k \in_\Pi \mathcal{AE}$, and $h_{\mathcal{A}_{\mathbf{w}}}(s) = \sum_{i=1}^k cost(\alpha_i(s), S_i^G)$. Now, for each $1 \leq i \leq k$ we have

$$\max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} d(G_i) = cost(\alpha_i(s), S_i^G),$$

and from disjointness of the constraints for $1 \leq i \neq i' \leq k$, we obtain

$$\max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} f(\mathbf{x}) = \max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} \sum_{i=1}^k d(G_i) = \sum_{i=1}^k \max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} d(G_i) = \sum_{i=1}^k cost(\alpha_i(s), S_i^G) = h_{\mathcal{A}_{\mathbf{w}}}(s).$$

From Definition 19 we thus have $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw}) \rangle$ being an LP-encoding of $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^k$ with respect to s. In turn, by Definition 20, $\sum_{i=1}^k ||\mathcal{T}_i|| = O(poly(||\Pi||))$, and hence $\mathbb{L}$ is both compact and poly-time constructible, finalizing the proof of the claim. ∎

5.3 LP-Optimization and Implicit Abstractions I: Fork Decomposition

Explicit abstractions are clearly the most well-known abstractions in the context of heuristic search in general, and planning as heuristic search in particular. Recently, however, Katz and Domshlak (2008c, 2009b) introduced the concept of *implicit abstraction heuristics* corresponding to certain *implicit abstractions* of planning tasks at hand. The basic idea behind implicit abstractions is in abstracting the task at hand into instances of provably tractable fragments of optimal planning. This, in particular, abolishes the limitation of explicit abstractions to use fixed-size abstract spaces only. Concretizing the idea of implicit abstractions, Katz and Domshlak (2008c) introduced a concrete framework based on decomposing the planning task along its causal graph and proposed a concrete instance of this framework, called *fork-decomposition*. Fork-decomposition (additive) abstraction is based on two specific fragments of tractable cost-optimal planning. For the technical details of fork decomposition in full we refer the reader to Katz and Domshlak (2008c). For our purposes here, it suffices to say that a fork decomposition of a planning task $\Pi = \langle V, A, I, G, cost \rangle$ over states S is obtained as follows.

(1) Schematic construction of a set of projection abstractions $\mathbf{\Pi} = \{\Pi^{\alpha_v^{\mathsf{f}}}, \Pi^{\alpha_v^{\mathsf{i}}}\}_{v \in V}$ with each $V^{\alpha_v^{\mathsf{f}}} = \{v\} \cup \mathsf{succ}(v)$ and $V^{\alpha_v^{\mathsf{i}}} = \{v\} \cup \mathsf{pred}(v)$. Note that, unlike explicit abstractions, each abstract space $S^{\alpha_v^{\mathsf{f}}}$ and $S^{\alpha_v^{\mathsf{i}}}$ can be of size $\Theta(|S|)$.

(2) Reformulation of the actions within the abstractions to single-effect actions only, so that the causal graphs of $\Pi^{\alpha_v^{\mathsf{f}}}$ and $\Pi^{\alpha_v^{\mathsf{i}}}$ become, respectively, "forks" and "inverted forks" rooted in v. After this action reformulation, the individual abstractions may cease being purely homomorphic.

(3) Within each $\Pi^{\alpha_v^{\mathsf{f}}}$, arbitrary abstraction of the domain of v to $\{0, 1\}$, and within each $\Pi^{\alpha_v^{\mathsf{i}}}$, arbitrary abstraction of the domain of v to $\{0, 1, \ldots, k\}$ with $k = O(1)$.

This decomposition of Π provides us with the **fork-decomposition ABS-ensemble** $\mathcal{AE} = \{\langle \mathcal{T}(\Pi^{\alpha^{\mathsf{f}}_v}), \alpha^{\mathsf{f}}_v, \beta^{\mathsf{f}}_v\rangle, \langle \mathcal{T}(\Pi^{\alpha^{\mathsf{i}}_v}), \alpha^{\mathsf{i}}_v, \beta^{\mathsf{i}}_v\rangle\}_{v\in V}$ of $\mathcal{T}(\Pi)$, with the action associations $\beta^{\mathsf{f}}_v, \beta^{\mathsf{i}}_v$ being established along the above steps (1-3). The additive abstractions of such an ABS-ensemble $\mathcal{AE}$ are of interest because (i) they can provide very informative heuristic estimates (Katz & Domshlak, 2009b), and (ii) each abstract task in $\mathbf{\Pi} = \{\Pi^{\alpha^{\mathsf{f}}_v}, \Pi^{\alpha^{\mathsf{i}}_v}\}_{v\in V}$ can be solved in polynomial time by special-purpose algorithms for the corresponding fragments of SAS$^+$ (Katz & Domshlak, 2008c). However, here as well, the choice of abstraction $\mathcal{A} \in_\Pi \mathcal{AE}$ with respect to $\mathcal{AE}$ is important, and optimizing this choice is clearly desirable. Interestingly, LP-optimization can come to the rescue here as well, despite our inability to perform an explicit search of TG-structures of $\mathcal{AE}$ in polynomial time.

Theorem 18 *For any planning task Π, any fork-decomposition ABS-ensemble of $\mathcal{T}(\Pi)$ is LP-optimizable, and thus $h_{\mathcal{AE}}(s)$ is poly-time computable for every state s of Π.*

Proof: From Lemmas 6 and 7 below, for each $v \in V$, we have each $\mathcal{AE}^{\mathsf{f}}_v = \{\langle \mathcal{T}(\Pi^{\alpha^{\mathsf{f}}_v}), \alpha^{\mathsf{f}}_v, \beta^{\mathsf{f}}_v\rangle\}$ and each $\mathcal{AE}^{\mathsf{i}}_i = \{\langle \mathcal{T}(\Pi^{\alpha^{\mathsf{i}}_v}), \alpha^{\mathsf{i}}_v, \beta^{\mathsf{i}}_v\rangle\}$ being LP-optimizable. From the composition Theorem 16, we thus have $\mathcal{AE}$ being LP-optimizable. ∎

Lemma 6 *For any planning task $\Pi = \langle V, A, I, G, cost\rangle$ and any variable $r \in V$, the "single fork" ABS-ensemble $\mathcal{AE}^{\mathsf{f}}_r = \{\langle \mathcal{T}(\Pi^{\alpha^{\mathsf{f}}_r}), \alpha^{\mathsf{f}}_r, \beta^{\mathsf{f}}_r\rangle\}$ of $\mathcal{T}(\Pi)$ is LP-optimizable.*

Proof: Given a planning task $\Pi = \langle V, A, I, G, cost\rangle$ and a variable $r \in V$, let $\mathcal{AE}^{\mathsf{f}}_r = \{\langle \mathcal{T}(\Pi^{\alpha^{\mathsf{f}}_r}), \alpha^{\mathsf{f}}_r, \beta^{\mathsf{f}}_r\rangle\}$ be a fork-decomposition of $\mathcal{T}(\Pi)$ over a single fork rooted in a binary domain variable r, and s be some state of Π. Our LP-encoding of such a "single fork" ABS-ensemble $\mathcal{AE}^{\mathsf{f}}_r$ with respect to s corresponds to an LP reformulation of the polynomial-time algorithm of Katz and Domshlak (2008c) for fork-structured tasks with binary root domain.

To simplify the notation, let us denote the variables $V^{\alpha^{\mathsf{f}}_r}$ of our fork-structured abstract task by V', its actions by A', its goal state by G', and the abstraction $\alpha^{\mathsf{f}}_r(s)$ of the state s in question by s'. We can assume that $G'[v]$ is defined for all $v \in V' \setminus \{r\}$; all the goal-less leaves can simply be omitted from the fork. For compliance with the notation of Katz and Domshlak (2008c), we denote the (abstracted to binary-valued) domain of r by $\mathcal{D}(r) = \{0, 1\}$ such that $s'[r] = 0$. Let $\sigma(r)$ be a 0/1 sequence of length $1 + \max_{v\in V'} |\mathcal{D}(v)|$, and, for $1 \le i \le |\sigma(r)|$, $\sigma(r)[i] = 0$ if i is odd, and $= 1$, if i is even. Let $\trianglerighteq^*[\sigma(r)]$ be the set of all nonempty prefixes of $\sigma(r)$ if $G'[r]$ is unspecified; otherwise let it be the set of all prefixes of $\sigma(r)$ ending with $G'[r]$. In what follows, for each of the two root's values $\vartheta \in \mathcal{D}(r)$, $\neg\vartheta$ denotes the opposite value $1 - \vartheta$. Let $DTG^{\vartheta_r}_v$ be the subgraph of $DTG(v, \Pi^{\alpha^{\mathsf{f}}_r})$ obtained by removing from the latter all the arcs labeled with $r = \neg\vartheta_r$.

To facilitate the presentation, the algorithm for fork-structured tasks $\Pi' = \langle V', A', s', G', cost'\rangle$ with binary root domain (Katz & Domshlak, 2008c) appears here as it appeared in the proof of Theorem 11.

(i) For each of the two values $\vartheta_r \in \mathcal{D}(r)$ of the root variable, each leaf variable $v \in V' \setminus \{r\}$, and each pair of values $\vartheta, \vartheta' \in \mathcal{D}(v)$, let $p_{\vartheta,\vartheta';\vartheta_r}$ be the cost of the cheapest sequence of actions ϑ to ϑ' *provided* $r = \vartheta_r$. The whole set $\{p_{\vartheta,\vartheta';\vartheta_r}\}$ can be computed by a straightforward variant of the all-pairs-shortest-paths Floyd-Warshall algorithm on $DTG^{\vartheta_r}_v$ in time $O(d^3|V|)$.

(ii) For each leaf $v \in V' \setminus \{r\}$, $1 \le i \le d + 1$, and $\vartheta \in \mathcal{D}(v)$, let $g_{\vartheta;i}$ be the cost of the cheapest sequence of actions changing $s'[v]$ to ϑ *provided* a sequence $\sigma \in \trianglerighteq^*[\sigma(r)]$, $|\sigma| = i$, of value

changes of r. Given $\{p_{\vartheta,\vartheta';\vartheta_r}\}$ as in (i), the set $\{g_{\vartheta;i}\}$ is given by the solution of the recursive equation

$$g_{\vartheta;i} = \begin{cases} p_{s'[v],\vartheta;s'[r]}, & i = 1 \\ \min\limits_{\vartheta'} g_{\vartheta';i-1} + p_{\vartheta',\vartheta;s'[r]}, & 1 < i \leq \delta_\vartheta,\ i \text{ is odd} \\ \min\limits_{\vartheta'} g_{\vartheta';i-1} + p_{\vartheta',\vartheta;\neg s'[r]}, & 1 < i \leq \delta_\vartheta,\ i \text{ is even} \\ g_{\vartheta;i-1}, & \delta_\vartheta < i \leq d+1 \end{cases},$$

where $\delta_\vartheta = |\mathcal{D}(v)| + 1$.

(iii) Given that,

$$h^*(s') = \min_{\sigma \in \trianglerighteq^*[\sigma(r)]} \left[cost'(\sigma) + \sum_{v \in V' \setminus \{r\}} g_{G'[v];|\sigma|} \right].$$

Our LP-encoding reformulates the algorithm as follows. First, we set the label-cost variables $\overrightarrow{w}$ to contain a variable w_a for each abstract action $a \in A'$; the additivity constraints $\mathbb{C}^{\mathsf{add}}(\overrightarrow{w})$ are defined in terms of these label-cost variables via β^{f}_r as in Eq. 5.3. Now we specify the LP-encoding $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw}) \rangle$ of $\mathcal{AE}^{\mathsf{f}}_r$ with respect to s as follows. The variable set $\overrightarrow{x}$ of $\mathbb{L}$ consists of three types of variables, notably

$$\overrightarrow{x} = \{h^{\mathsf{f}}\} \cup \bigcup_{\substack{v \in V' \setminus \{r\}, \\ \vartheta \in \mathcal{D}(v), \\ 1 \leq i \leq |\sigma(r)|}} \{d(v,\vartheta,i)\} \cup \bigcup_{\substack{v \in V' \setminus \{r\}, \\ \vartheta,\vartheta' \in \mathcal{D}(v), \\ \vartheta_r \in \{0,1\}}} \{p(v,\vartheta,\vartheta',\vartheta_r)\}.$$

- The variable h^{f} stands for the minimal cost of solving our fork-structured planning task, and the maximized objective function of $\mathbb{L}$ is simply $f(\overrightarrow{x}) = h^{\mathsf{f}}$.
- Each variable $d(v,\vartheta,i)$ stands for the cost of the cheapest sequence of actions affecting v that changes its value from $s'[v]$ to ϑ *given* that the value changes of r induce a 0/1 sequence of r's values of length i.
- Each variable $p(v,\vartheta,\vartheta',\vartheta_r)$ stands for the cost of the cheapest sequence of actions affecting v that changes its value from ϑ to ϑ' having *fixed* the value of r to ϑ_r.

The constraint set $\mathbb{C}^{\mathcal{AE}}$ of $\mathbb{L}$ consists of the following sets of linear constraints.

(i) For each $v \in V' \setminus \{r\}$, $\vartheta \in \mathcal{D}(v)$,

$$p(v,\vartheta,\vartheta,0) = 0, \qquad p(v,\vartheta,\vartheta,1) = 0.$$

Likewise, for each v-changing action $a \in A'$ and each $\vartheta_r \in \mathsf{Pre}(a)[r]$, if $v \in \mathcal{V}(\mathsf{pre}(a))$, then

$$p(v,\vartheta,\mathsf{eff}(a)[v],\vartheta_r) \leq p(v,\vartheta,\mathsf{pre}(a)[v],\vartheta_r) + w_a$$

and otherwise,

$$p(v,\vartheta,\mathsf{eff}(a)[v],\vartheta_r) \leq w_a.$$

Semantics: Shortest-path constraints as in Eq. 5.6.

(ii) For each leaf $v \in V' \setminus \{r\}$ and each $\vartheta \in \mathcal{D}(v)$

$$d(v, \vartheta, 1) \leq p(v, s'[v], \vartheta, \sigma(r)[1])$$

and, for each $\vartheta' \in \mathcal{D}(v)$, and $1 < i \leq |\sigma(r)|$,

$$d(v, \vartheta, i) \leq d(v, \vartheta', i-1) + p(v, \vartheta', \vartheta, \sigma(r)[i])$$

Semantics: The cost of achieving ϑ from $s'[v]$ given $|\sigma(r)| = i$ is bounded by the cost of achieving ϑ' given $|\sigma(r)| = i - 1$, and achieving ϑ from ϑ' given $\sigma(r)[i]$.

(iii) For all goal-achieving sequences $\sigma \in \trianglerighteq^*[\sigma(r)]$ of value changes of r, and each pair of r-changing actions $a, a' \in A'$ such that $\mathsf{eff}(a)[r] = 1$ and $\mathsf{eff}(a')[r] = 0$,

$$h^{\mathsf{f}} \leq \lceil \frac{|\sigma| - 1}{2} \rceil \cdot w_a + \lfloor \frac{|\sigma| - 1}{2} \rfloor \cdot w_{a'} + \sum_{v \in V' \setminus \{r\}} d(v, G'[v], |\sigma|)$$

Semantics: The cost of solving the task is not greater than the sum of achieving the goal values for all the leaves given a value sequence of the root, plus the cost of providing that value sequence.

This finalizes our LP-encoding for an ABS-ensemble consisting of a single fork-structured abstraction with a binary root domain. Let $\mathbf{w}$ be an arbitrary point in $\mathscr{H}^{\mathsf{add}}$ and let the action cost function $cost'$ be set according to $\mathbf{w}$. That is, for each $a \in A'$, $cost'(a) = \mathbf{w}[w_a]$. Let vector $\mathbf{x} \in \mathcal{D}(\overrightarrow{x})$ be specified according to the values obtained while running Katz and Domshlak's algorithm on $\Pi' = \langle V', A', s', G', cost' \rangle$ as follows.

$$\begin{aligned} \mathbf{x}[p(v, \vartheta, \vartheta', \vartheta_r)] &= p_{\vartheta, \vartheta'; \vartheta_r}, \\ \mathbf{x}[d(v, \vartheta, i)] &= g_{\vartheta; i} \\ \mathbf{x}[h^{\mathsf{f}}] &= \min_{\sigma \in \trianglerighteq^*[\sigma(r)]} \left[cost'(\sigma) + \sum_{v \in V' \setminus \{r\}} g_{G'[v]; |\sigma|} \right]. \end{aligned}$$

Note that constraint sets (i), (ii), and (iii), respectively, reformulate steps (i), (ii), and (iii) of the algorithm while keeping the action costs as free variables $\{w_a\}$. Hence, $\mathbf{xw}$ agrees with the constraint sets (i), (ii), and (iii), and thus $\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}$, implying

$$\max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} h^{\mathsf{f}} \geq \mathbf{x}[h^{\mathsf{f}}] = h^*(s').$$

For the other direction, let $\mathbf{x} \in \mathcal{D}(\overrightarrow{x})$ be such that $\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}$, $\sigma \in \trianglerighteq^*[\sigma(r)]$ be the one responsible for the minimum found by the algorithm for the planning task $\Pi' = \langle V', A', s', G', cost' \rangle$, and let ρ be an optimal plan for Π' such that $\rho{\downarrow_r}$ induces σ. For each $v \in V' \setminus \{r\}$, let $\rho{\downarrow_v} = \rho_1, \ldots, \rho_m$, $1 \leq m \leq |\sigma|$ be the sequence of actions changing the values of v, split by execution of r-changing actions $\rho{\downarrow_r}$ along ρ. Each such $\rho_i = \langle a^i_1 \ldots a^i_{k_i} \rangle$ is a sequence of actions inducing a sequence of v-values $\vartheta^i_0, \vartheta^i_1, \ldots, \vartheta^i_{k_i}$, and $\vartheta^1_0, \vartheta^1_1, \ldots, \vartheta^1_{k_1}, \vartheta^2_1, \ldots, \vartheta^2_{k_2}, \ldots, \vartheta^m_1, \ldots, \vartheta^m_{k_m}$ is a simple path from $s'[v] = \vartheta^1_0$ to $G'[v] = \vartheta^m_{k_m}$ in the domain transition graph of v such that, for $1 \leq i \leq m$,

the change to ϑ^i_j, $1 \leq j \leq k_i$ is labeled by either $\sigma[i]$ or nothing at all. For ease of presentation, for each $1 \leq i \leq m$ we denote $\vartheta^{i+1}_0 = \vartheta^i_{k_i}$. Given that,

$$\mathbf{x}[d(v, G'[v], |\sigma|)] \overset{(ii)}{\leq} \sum_{i=1}^{m} \sum_{j=1}^{k_i} \mathbf{x}[p(v, \vartheta^i_{j-1}, \vartheta^i_j, \sigma[i])] \overset{(i)}{\leq} \sum_{i=1}^{m} \sum_{j=1}^{k_i} \mathbf{w}[a^i_j] = cost'(\rho{\downarrow_v}),$$

and thus

$$\mathbf{x}[h^{\mathsf{f}}] \overset{(iii)}{\leq} \sum_{a \in \rho{\downarrow_r}} \mathbf{w}[a] + \sum_{v \in V' \setminus \{r\}} \mathbf{x}[d(v, G'[v], |\sigma|)] \leq cost'(\rho{\downarrow_r}) + \sum_{v \in V' \setminus \{r\}} cost'(\rho{\downarrow_v}) = cost'(\rho),$$

implying

$$\max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} h^{\mathsf{f}} \leq h^*(s').$$

Putting the two directions together, for each $\mathbf{w} \in \mathscr{H}^{\mathsf{add}}$, we have

$$\max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} f(\mathbf{x}) = h_{\mathcal{A}_{\mathbf{w}}}(s),$$

and from Definition 19 we thus have $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw}) \rangle$ being an LP-encoding of $\mathcal{AE}^{\mathsf{f}}_r$. Note that if $D = \max_{v \in V'} |\mathcal{D}(v)|$, then the LP variable set size $|\overrightarrow{x}|$ is $O(|V| \cdot D^2)$ and the constraint set size $|\mathbb{C}^{\mathcal{AE}}|$ is $O(D \cdot |A'|^2 + |V'| \cdot D^3 + |A'| \cdot |V'| \cdot D)$, and thus the LP-encoding is compact.

■

Lemma 7 *For any planning task $\Pi = \langle V, A, I, G, cost \rangle$ and any variable $r \in V$, the "single inverted fork" ABS-ensemble $\mathcal{AE}^{\mathsf{i}}_r = \{\langle \mathcal{T}(\Pi^{\alpha^{\mathsf{i}}_r}), \alpha^{\mathsf{i}}_r, \beta^{\mathsf{i}}_r \rangle\}$ of $\mathcal{T}(\Pi)$ is LP-optimizable.*

Proof: Given a planning task $\Pi = \langle V, A, I, G, cost \rangle$ and a variable $r \in V$, let $\mathcal{AE}^{\mathsf{i}}_r = \{\langle \mathcal{T}(\Pi^{\alpha^{\mathsf{i}}_r}), \alpha^{\mathsf{i}}_r, \beta^{\mathsf{i}}_r \rangle\}$ be a fork-decomposition of $\mathcal{T}(\Pi)$ over a single inverted fork with sink r, and let s be some state of Π. Our LP-encoding of such a "single inverted fork" ABS-ensemble $\mathcal{AE}^{\mathsf{i}}_r$ with respect to state s corresponds to an LP reformulation of the polynomial-time algorithm of Katz and Domshlak (2008c) for inverted-fork structured tasks with root domain of size $O(1)$.

As in the proof of Lemma 6, let us denote the variables $V^{\alpha^{\mathsf{i}}_r}$ of our inverted-fork structured abstract task by V', its actions by A', its goal state by G', and the abstraction $\alpha^{\mathsf{i}}_r(s)$ of the state s in question by s'. First, we set the label-cost variables $\overrightarrow{w}$ to contain a variable w_a for every abstract action $a \in A'$; the additivity constraints $\mathbb{C}^{\mathsf{add}}(\overrightarrow{w})$ are then defined in terms of these label-cost variables via β^{i} as in Eq. 5.3. To facilitate the presentation, the algorithm for inverted-fork structured tasks $\Pi' = \langle V', A', s', G', cost' \rangle$ with sink domain of size $O(1)$ (Katz & Domshlak, 2008c) appears here as it appeared in the proof of Theorem 12.

(i) For each parent variable $v \in V' \setminus \{r\}$, and each pair of its values $\vartheta, \vartheta' \in \mathcal{D}(v)$, let $p_{\vartheta,\vartheta'}$ be the cost of the cheapest sequence of actions changing ϑ to ϑ'. The whole set $\{p_{\vartheta,\vartheta'}\}$ can be computed using the Floyd-Warshall algorithm on the domain transition graph of v in time $O(d^3|V|)$.

(ii) For each cycle-free path $\pi = a_1 \cdot \ldots \cdot a_m$ from $s'[r]$ to $G'[r]$ in $DTG(r, \Pi')$, let g_π be the cost of the cheapest plan from s' in Π' based on π, and the shortest paths computed in (1). Each g_π can be computed as

$$g_\pi = \sum_{i=1}^{m} cost(a_i) + \sum_{i=0}^{m} \sum_{v \in V' \setminus \{r\}} p_{\mathsf{pre}_i[v], \mathsf{pre}_{i+1}[v]},$$

where $\{\mathsf{pre}_0 \cdot \ldots \cdot \mathsf{pre}_{m+1}\}$ are the values needed from the parents of r along the path π. That is, for each $v \in V' \setminus \{r\}$, and $0 \leq i \leq m+1$,

$$\mathsf{pre}_i[v] = \begin{cases} s'[v], & i = 0 \\ G'[v], & i = m+1, \text{ and } G'[v] \text{ is specified} \\ \mathsf{pre}(a_i)[v], & 1 \leq i \leq m, \text{ and } \mathsf{pre}(a_i)[v] \text{ is specified} \\ \mathsf{pre}_{i-1}[v] & \text{otherwise} \end{cases}. \tag{5.8}$$

From that, we have $h^*(s') = \min_\pi g_\pi$.

Given that any optimal plan for Π' must correspond to a cycle-free path in $DTG(r, \Pi')$ from $s'[r]$ to $G'[r]$, our LP-encoding $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw}) \rangle$ of $\mathcal{AE}$ with respect to state s reformulates the algorithm as follows. The variable set $\overrightarrow{x}$ of $\mathbb{L}$ consists of two types of variables, notably

$$\overrightarrow{x} = \{h^{\mathsf{i}}\} \cup \bigcup_{\substack{v \in V' \setminus \{r\}, \\ \vartheta, \vartheta' \in \mathcal{D}(v)}} \{d(v, \vartheta, \vartheta')\}.$$

- The variable h^{i} stands for the minimal cost of solving our inverted fork-structured planning task, and the maximized objective function of $\mathbb{L}$ is simply $f(\overrightarrow{x}) = h^{\mathsf{i}}$.
- Each variable $d(v, \vartheta, \vartheta')$ stands for the cost of the cheapest sequence of actions affecting v that changes its value from ϑ to ϑ'.

The constraint $\mathbb{C}^{\mathcal{AE}}$ of $\mathbb{L}$ consists of the following sets of linear constraints:

(i) For each $v \in V' \setminus \{r\}$ and each $\vartheta \in \mathcal{D}(v)$,

$$d(v, \vartheta, \vartheta) = 0.$$

Likewise, for each v-changing action $a \in A'$, if $v \in \mathcal{V}(\mathsf{pre}(a))$, then

$$d(v, \vartheta, \mathsf{eff}(a)[v]) \leq d(v, \vartheta, \mathsf{pre}(a)[v]) + w_a$$

and otherwise,

$$d(v, \vartheta, \mathsf{eff}(a)[v]) \leq w_a.$$

Semantics: Shortest-path constraints as in Eq. 5.6.

(ii) For each cycle-free path $\pi = a_1 \cdot \ldots \cdot a_m$ from $s'[r]$ to $G'[r]$ in $DTG(r, \Pi')$,

$$h^{\mathsf{i}} \leq \sum_{i=1}^{m} w_{a_i} + \sum_{i=0}^{m} \sum_{v \in V' \setminus \{r\}} d(v, \mathsf{pre}_i[v], \mathsf{pre}_{i+1}[v])$$

where, for each $0 \leq i \leq m+1$, pre_i is defined as in Eq. 5.8.

Semantics: The cost of solving the task is not greater than the cost of any cycle-free path of r plus sums of costs of reaching the prevail conditions of actions on this path and reaching the goal afterwards.

This finalizes our LP-encoding for an ABS-ensemble consisting of a single inverted-fork structured abstraction with a $O(1)$-bounded root domain. Let $\mathbf{w}$ be an arbitrary point in $\mathscr{H}^{\mathsf{add}}$ and let the action cost function $cost'$ be set according to $\mathbf{w}$. That is, for each $a \in A'$, $cost'(a) = \mathbf{w}[w_a]$. Let vector $\mathbf{x} \in \mathcal{D}(\overrightarrow{x})$ be specified according to the values obtained while running Katz and Domshlak's algorithm on $\Pi' = \langle V', A', s', G', cost' \rangle$ as follows.

$$\mathbf{x}[d(v, \vartheta, \vartheta')] = p_{\vartheta,\vartheta'},$$
$$\mathbf{x}[h^{\mathsf{i}}] = \min_{\pi} g_{\pi}.$$

Note that constraint sets (i) and (ii), respectively, reformulate steps (i) and (ii) of the algorithm while keeping the action costs as free variables $\{w_a\}$. Hence, $\mathbf{xw}$ agrees with the constraint sets (i) and (ii), and thus $\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}$, implying

$$\max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} h^{\mathsf{i}} \geq \mathbf{x}[h^{\mathsf{i}}] = h^*(s').$$

For the other direction, let $\mathbf{x} \in \mathcal{D}(\overrightarrow{x})$ be such that $\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}$, and let ρ be an optimal plan for the planning task $\Pi' = \langle V', A', s', G', cost' \rangle$. Let $\pi = a_1 \cdot \ldots \cdot a_m$ be the path from $s'[r]$ to $G'[r]$ in $DTG(r, \Pi')$ induced by ρ. Note that π is cycle-free by the optimality of ρ. For each $v \in V' \setminus \{r\}$, let $\rho{\downarrow_v} = \rho_0, \ldots, \rho_m$ be the sequence of actions changing the values of v, split by execution of r-changing actions of π along ρ. Then each $\rho_i = \langle a_1^i \ldots a_{k_i}^i \rangle$ is a sequence of actions changing the value of v from $\mathsf{pre}_i[v]$ to $\mathsf{pre}_{i+1}[v]$ as in Eq 5.8, and

$$\sum_{i=0}^{m} \mathbf{x}[d(v, \mathsf{pre}_i[v], \mathsf{pre}_{i+1}[v])] \overset{\text{(i)}}{\leq} \sum_{i=0}^{m} \sum_{j=1}^{k_i} \mathbf{w}[w_{a_j^i}] = cost'(\rho{\downarrow_v}).$$

Hence,

$$\mathbf{x}[h^{\mathsf{i}}] \overset{\text{(ii)}}{\leq} \sum_{i=0}^{m} \mathbf{w}[w_{a_i}] + \sum_{v \in V' \setminus \{r\}} \sum_{i=0}^{m} \mathbf{x}[d(v, \mathsf{pre}_i[v], \mathsf{pre}_{i+1}[v])] \leq cost'(\pi) + \sum_{v \in V' \setminus \{r\}} cost'(\rho{\downarrow_v}) = cost'(\rho) = h^*(s'),$$

implying

$$\max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} h^{\mathsf{i}} \leq h^*(s').$$

Putting the two directions together, for each $\mathbf{w} \in \mathscr{H}^{\mathsf{add}}$, we have

$$\max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} f(\mathbf{x}) = h_{\mathcal{A}_{\mathbf{w}}}(s),$$

and from Definition 19 we thus have $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw}) \rangle$ being an LP-encoding of $\mathcal{AE}_r^{\mathsf{i}}$. Note that, if $D = \max_{v \in V'} |\mathcal{D}(v)|$ and $d = |\mathcal{D}(r)|$, then the LP variable set size $|\overrightarrow{x}|$ is $O(|V'| \cdot D^2)$ and the constraint set size $|\mathbb{C}^{\mathcal{AE}}|$ is $O(d^d + |V'| \cdot |A'| \cdot D)$, and thus the LP-encoding is compact and $\mathcal{AE}_r^{\mathsf{i}}$ is LP-optimizable.

■

5.4 LP-Optimization and Implicit Abstractions II: Tree-structured COPs

Fork-decomposition implicit abstractions are grounded in two specific fragments of tractable cost-optimal planning. In principle, however, implicit abstractions based on some other tractable fragments might also lead to LP-optimizable ABS-ensembles. Here we consider two such fragments recently characterized by Katz and Domshlak (2008a), namely $\mathbf{P}_b$ and $\mathbf{P}(1)$ (see Section 3.1).

Katz and Domshlak's poly-time algorithms for the planning fragments $\mathbf{P}_b$ and $\mathbf{P}(1)$ differ substantially. However, both correspond to *reductions of the planning task to compact, tree-structured constraint optimization problems (COPs)*. This joint property of $\mathbf{P}_b$ and $\mathbf{P}(1)$ (that might also hold for some other interesting planning fragments) facilitates our objective of adding $\mathbf{P}_b$- and $\mathbf{P}(1)$-based implicit abstractions to the "$h_{\mathcal{AE}}$-friendly" family of LP-optimizable ABS-ensembles.

Let us start by considering a general, tree-structured constraint optimization problem $\mathsf{COP} = (\mathcal{X}, \mathcal{F})$ over finite-domain variables $\mathcal{X}$, functional components $\mathcal{F}$, and the objective $\min \sum_{\varphi \in \mathcal{F}} \varphi(\mathcal{X})$. Fixing an arbitrary rooting of the COP's constraint network at some $r \in \mathcal{X}$, in what follows we refer to that rooted tree of COP via its set of *directed* edges $E = \{(x, x')\}$, oriented "from the root to the leaves." In these terms, we have

$$\mathcal{F} = \{\varphi_x : \mathcal{D}(y) \times \mathcal{D}(x) \to \mathbb{R}^{0+} \mid (y, x) \in E\}.$$

It is well known that tree-structured COPs as above can be solved in low polynomial time by a dynamic-programming-style, message-passing algorithm (Dechter, 2003). But like the Dijkstra and breadth-first search algorithms for solving explicit abstractions, this message-passing algorithm does not appear to meet our needs. The good news, however, is that such tree-structured COPs can also be solved via linear programming. While we suspect that this quite straightforward, LP formulation is not new, we found no previous mention of it in the literature.

Given a tree-structured constraint optimization problem $\mathsf{COP} = (\mathcal{X}, \mathcal{F})$, its LP formulation is using a set of nonnegative, real-valued variables

$$\overrightarrow{c} = \{h^{\mathsf{cop}}\} \cup \bigcup_{\substack{(x',x) \in E, \\ \overline{x}' \in \mathcal{D}(x')}} \{c(x|\overline{x}')\},$$

with the semantics of each variable $c(x|\overline{x}')$ being an "optimal solution for the subtree rooted at x given that the parent x' of x takes the value $\overline{x}'$". The actual linear program is then

$$\begin{aligned} &\max_{\overrightarrow{c}} \ h^{\mathsf{cop}} \\ &\text{s.t. } \forall \overline{r} \in \mathcal{D}(r): \quad h^{\mathsf{cop}} \leq \sum_{(r,x) \in E} c(x|\overline{r}), \\ &\qquad \forall (x,y) \in E, \overline{x} \in \mathcal{D}(x), \overline{y} \in \mathcal{D}(y): \\ &\qquad\qquad c(y|\overline{x}) \leq \sum_{(y,z) \in E} c(z|\overline{y}) + \varphi_y(\overline{x}, \overline{y}), \end{aligned} \tag{5.9}$$

and its solution induces a solution for COP.

Lemma 8 *Given a tree-structured constraint optimization problem* $\mathsf{COP} = (\mathcal{X}, \mathcal{F})$ *over finite-domain variables* $\mathcal{X}$ *and functional components* $\mathcal{F}$*, we have*

$$\min_{\overline{x} \in \mathcal{D}(\mathcal{X})} \sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}) = \max_{\mathbf{c} \in \mathscr{H}^{\mathsf{COP}}} h^{\mathsf{cop}}$$

where $\mathscr{H}^{\mathsf{COP}}$ *is the convex polyhedron specified by the linear constraints as in Eq. 5.9.*

Proof: First, note that

$$\forall \overline{x} \in \mathcal{D}(\mathcal{X}) \text{ and } \forall \mathbf{c} \in \mathscr{H}^{\mathsf{COP}}, \;\; h^{\mathsf{cop}} \leq \sum_{(y,z)} \varphi_z(\overline{x}) = \sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}).$$

Therefore

$$\max_{\mathbf{c} \in \mathscr{H}^{\mathsf{COP}}} h^{\mathsf{cop}} \leq \min_{\overline{x} \in \mathcal{D}(\mathcal{X})} \sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}).$$

Let $\mathbf{c} \in \mathscr{H}^{\mathsf{COP}}$ be an assignment constructed by going over the edges (y, z) of the tree bottom up, and for each $\overline{y} \in \mathcal{D}(y)$ setting

$$c(z|\overline{y}) = \min_{\overline{z} \in \mathcal{D}(z)} \sum_{(z,z')} c(z'|\overline{z}) + \varphi_z(\overline{y}, \overline{z}) \tag{5.10}$$

and then setting

$$h^{\mathsf{cop}} = \min_{\overline{r} \in \mathcal{D}(r)} \sum_{(r,x)} c(x|\overline{r}). \tag{5.11}$$

Let $\overline{x} \in \mathcal{D}(\mathcal{X})$ be the combination of assignments to COP variables that establish the minimum in Eq. 5.10-5.11. Then

$$h^{\mathsf{cop}} = \min_{\overline{x} \in \mathcal{D}(\mathcal{X})} \sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}),$$

and therefore

$$\max_{\mathbf{c} \in \mathscr{H}^{\mathsf{COP}}} h^{\mathsf{cop}} \geq \min_{\overline{x} \in \mathcal{D}(\mathcal{X})} \sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}).$$

■

With Lemma 8 at hand, we now take two additional steps towards an LP-encoding of ABS-ensembles $\mathcal{AE}$ containing implicit abstractions reducible to tree-structured COPs. In each such *individual* implicit abstraction, each value $\varphi_y(\overline{x}, \overline{y})$ of each functional component φ_y must be *somehow* precomputed from the costs of the planning actions. In our case, however, the costs of the actions in the abstract tasks are not fixed in advance, but should be established by the LP-optimization process. In what follows, we consider this matter more closely.

Given a planning task Π, let $\mathcal{AE} = \{\langle \mathcal{T}(\Pi'), \alpha, \beta \rangle\}$ be a single-abstraction ABS-ensemble of $\mathcal{T}(\Pi)$ such that cost-optimal planning for Π' is reducible to a tree-structured constraint optimization problem $\mathsf{COP}_{\Pi'}$. First, *suppose* that, for any *fixed* vector of action costs $\mathbf{w}^\dagger$, each functional-component value $\overline{\varphi} \equiv \varphi_y(\overline{x}, \overline{y})$ corresponds to the solution value of some compact canonical-form linear program

$$\begin{aligned} &\max f_{\overline{\varphi}}(\overrightarrow{z_{\overline{\varphi}} w}) \\ &\text{s.t. } \mathbf{A}_{\overline{\varphi}} \cdot \overrightarrow{z_{\overline{\varphi}} w} \leq \mathbf{b}_{\overline{\varphi}} \\ &\qquad\qquad \overrightarrow{w} \leq \mathbf{w}^\dagger \end{aligned} \tag{5.12}$$

where $\mathbf{A}_{\overline{\varphi}}$ and $\mathbf{b}_{\overline{\varphi}}$ are a matrix and a vector of coefficients, respectively. If so, then, given $\mathbf{w}^\dagger$, we can reformulate the linear program in Eq. 5.9 by

(i) replacing the *constants* $\overline{\varphi} \equiv \varphi_y(\overline{x}, \overline{y})$ by the corresponding affine functions $f_{\overline{\varphi}}(\overrightarrow{z_{\overline{\varphi}} w})$, and

(ii) for each $\overline{\varphi}$, adding its linear constraints as in Eq. 5.12.

The extended program is still linear, and we still have

$$\min_{\overline{x} \in \mathcal{D}(\mathcal{X})} \sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}) = \max_{\mathbf{czw} \in \mathscr{H}^{\mathsf{COP}}} h^{\mathsf{cop}}$$

where $\mathbf{z}$ and $\mathbf{w}$ are assignments to $\overrightarrow{z} = \bigcup_{\overline{\varphi}} \overrightarrow{z}_{\overline{\varphi}}$ and action-cost variables $\overrightarrow{w}$, respectively, and $\mathscr{H}^{\mathsf{COP}}$ is the convex polyhedron specified by these *extended* linear constraints.

The extended linear program specified above for implicit abstraction Π' with $\mathsf{COP}_{\Pi'}$ satisfying Eq. 5.12 provides the basis for the LP-encoding of the corresponding ABS-ensembles $\mathcal{AE} = \{\langle \mathcal{T}(\Pi'), \alpha, \beta \rangle\}$. First, as before, let the label-cost variables $\overrightarrow{w}$ contain a variable w_a for every abstract action $a \in A'$ of Π'; the additivity constraints $\mathbb{C}^{\mathsf{add}}(\overrightarrow{w})$ are defined in terms of these label-cost variables via β as in Eq. 5.3. Now, given a state s of Π, we specify an LP-encoding $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw}) \rangle$ of $\mathcal{AE}$ with respect to state s as follows.

- The variable set $\overrightarrow{x} = \overrightarrow{cz}$ consists of the variables of Eqs. 5.9 and 5.12, and the objective of $\mathbb{L}$ is $f(\overrightarrow{x}) = h^{\mathsf{cop}}$.
- The constraint $\mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw})$ of $\mathbb{L}$ consists of all the linear constraints from Eq. 5.9, as well as the constraint $\mathbf{A}_{\overline{\varphi}} \cdot \overrightarrow{z_{\overline{\varphi}} w} \leq \mathbf{b}_{\overline{\varphi}}$ from Eq. 5.12 for each functional-component value $\overline{\varphi}$ of $\mathsf{COP}_{\Pi'}$.

This finalizes the desired LP-encoding; extending it to such multiple-abstraction ABS-ensembles $\mathcal{AE} = \{\langle \mathcal{T}(\Pi'_i), \alpha_i, \beta_i \rangle\}_{i=1}^k$ (and, again, possibly some other LP-optimizable abstractions) is ensured by the composition Theorem 16.

Theorem 19 *Given a planning task Π, and an ABS-ensemble $\mathcal{AE} = \{\langle \mathcal{T}(\Pi'_i), \alpha_i, \beta_i \rangle\}_{i=1}^k$ of $\mathcal{T}(\Pi)$, if cost-optimal planning for each Π'_i is poly-time reducible to a compact and tree-structured constraint optimization problem satisfying Eq. 5.12, and $k = O(poly(|\Pi|))$, then $\mathcal{AE}$ is LP-optimizable, and thus $h_{\mathcal{AE}}(s)$ is poly-time computable for every state $s \in S$.*

Proof: Let Π be a planning task, $\mathcal{AE} = \{\langle \mathcal{T}(\Pi'_i), \alpha_i, \beta_i \rangle\}_{i=1}^k$ be an ABS-ensemble of $\mathcal{T}(\Pi)$ as in the claim, s be some state of Π, and $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw}) \rangle$ be as described above. For any $\mathbf{w} \in \mathscr{H}^{\mathsf{add}}$, let $\mathcal{A}_{\mathbf{w}} = \{\langle \langle \mathcal{T}_i, \mathbf{w} \rangle, \alpha_i \rangle\}_{i=1}^k \in_\Pi \mathcal{AE}$, and therefore $h_{\mathcal{A}_{\mathbf{w}}}(s) = \sum_{i=1}^k cost(\alpha_i(s), S_i^G)$. Note that, since Π'_i with initial state $\alpha_i(s)$ is poly-time reducible to a compact and tree-structured constraint optimization problem $\mathsf{COP}_i = (\mathcal{X}_i, \mathcal{F}_i)$ satisfying Eq. 5.12, we have

$$cost(\alpha_i(s), S_i^G) = \min_{\overline{x} \in \mathcal{D}(\mathcal{X}_i)} \sum_{\varphi \in \mathcal{F}_i} \varphi(\overline{x}).$$

From Lemma 8 we then have

$$cost(\alpha_i(s), S_i^G) = \max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} h_i^{\mathsf{cop}},$$

and therefore

$$\max_{\mathbf{xw}\in\mathscr{H}^{\mathcal{AE}}} f(\mathbf{x}) = \sum_{i=1}^{k} cost(\alpha_i(s), S_i^G) = h_{\mathcal{A}_\mathbf{w}}(s).$$

From Definition 19 we then have $\mathbb{L} = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathbb{C}^{\mathcal{AE}}(\overrightarrow{xw})\rangle$ being an LP-encoding of $\mathcal{AE}$. In turn, since $k = O(poly(|\Pi|))$ and, for $1 \leq i \leq k$, cost-optimal planning for each abstract task Π'_i is poly-time reducible to a compact and tree-structured constraint optimization problem satisfying Eq. 5.12, the LP-encoding of $\mathcal{AE}$ is both compact and poly-time constructible for any state s of Π. Hence, $h_{\mathcal{AE}}(s)$ is poly-time computable for any state s of Π. ∎

Last but not least is, of course, the question of whether the requirement posed by Eq. 5.12 is relevant to the constraint optimization problems induced by the planning tasks from the known fragments of tractability. The good news is that Theorems 20 and 21 provide an affirmative answer to this question.

Theorem 20 *For any task $\Pi = \langle V, A, I, G, cost\rangle$ in $\mathbf{P}_b$, cost-optimal planning for Π is poly-time reducible to a compact and tree-structured constraint optimization problem satisfying Eq. 5.12.*

Proof: The proof is based on the algorithm of Katz and Domshlak (2008a) for constructing a constraint optimization problem $\mathsf{COP}_\Pi = (\mathcal{X}, \mathcal{F})$ over tree-structured constraint network, as presented in Section A.1 and depicted in Figure A.3. Let $\Pi = \langle V, A, I, G, cost\rangle$ be a planning task in $\mathbf{P}_b$, and let $\mathsf{COP}_\Pi = (\mathcal{X}, \mathcal{F})$ be the constraint optimization problem resulting in runnig the algorithm. For each $\varphi \in \mathcal{F}$ and each assignment to the scope of φ, we define a linear program satisfying Eq. 5.12 as follows. First, for each planning variable v with $\mathsf{pred}(v) = \emptyset$, and each of its goal-valid (time-stamped) value-changing sequences $\tau' \in \trianglerighteq^*[\tau(v)]$, the constraints set as in Eq. 5.12 corresponding to $\overline{\varphi} = \varphi_v(\tau')$ is

$$z_{\overline{\varphi}} \leq \left\lfloor \frac{|\tau'|}{2} \right\rfloor \cdot w_a + \left\lfloor \frac{|\tau'|-1}{2} \right\rfloor \cdot w_{a'},$$

for each $a, a' \in A_v$ such that $\mathsf{eff}(a)[v] = 1$ and $\mathsf{eff}(a')[v] = 0$. Next, for each planning variable v with $\mathsf{pred}(v) = \{u_1, \ldots, u_k\}, k \geq 1$, each goal-valid value-changing sequence $\tau' \in \trianglerighteq^*[\tau(v)]$ of v, and each set of such goal-valid value-changing sequences $\{\tau'_1 \in \trianglerighteq^*[\tau(u_1)], \ldots, \tau'_k \in \trianglerighteq^*[\tau(u_k)]\}$ of v's parents, the constraints set corresponding to $\overline{\varphi} = \varphi_v(\tau', \tau'_1, \ldots, \tau'_k)$ is the set of shortest-path constraints of the digraph $G'_e(v)$ as in Eq. 5.6. The union of all the constraint sets above together with the LP formulation as in Eq. 5.9 results in a linear program satisfying Eq. 5.12. ∎

Theorem 21 *For any task $\Pi = \langle V, A, I, G, cost\rangle$ in $\mathbf{P}(1)$, cost-optimal planning for Π is poly-time reducible to a compact and tree-structured constraint optimization problem satisfying Eq. 5.12.*

Proof: The proof is based on the algorithm of Katz and Domshlak (2008a) for constructing a constraint optimization problem $\mathsf{COP}_\Pi = (\mathcal{X}, \mathcal{F})$ over tree-structured constraint network, as presented in Section A.3 and depicted in Figure A.7. Let $\Pi = \langle V, A, I, G, cost\rangle$ be a planning task in $\mathbf{P}(1)$, and let $\mathsf{COP}_\Pi = (\mathcal{X}, \mathcal{F})$ be the constraint optimization problem resulting in runnig the algorithm. For each $\varphi \in \mathcal{F}$ and each assignment to the scope of φ, we define a linear program satisfying Eq. 5.12 as follows. The key points is that, for each $\varphi \in \mathcal{F}$ and for each assignment $\overline{\varphi}$ to the scope of φ, we can define a linear program satisfying Eq. 5.12 by exploiting the fact that

domain ($\mathcal{D}$)	$h^{\mathcal{F}}$					$h^{\mathcal{J}}$					$h^{\mathcal{FJ}}$				
	U	O	SB	$\frac{N(\mathrm{U})}{N(\mathrm{O})}$	$\frac{t(\mathrm{O})}{t(\mathrm{U})}$	U	O	SB	$\frac{N(\mathrm{U})}{N(\mathrm{O})}$	$\frac{t(\mathrm{O})}{t(\mathrm{U})}$	U	O	SB	$\frac{N(\mathrm{U})}{N(\mathrm{O})}$	$\frac{t(\mathrm{O})}{t(\mathrm{U})}$
airport-ipc4	11	7	7	1.08	151.11	14	15	14	31.83	16.64	11	7	7	1.45	141.94
blocks-ipc2	17	11	11	1.61	474.53	15	17	15	165.39	49.39	15	11	10	6.22	510.07
depots-ipc3	2	1	1	0.97	451.14	2	2	2	10.59	31.78	2	1	1	1.99	437.38
driverlog-ipc3	9	6	6	18.30	281.12	10	10	10	38.98	57.33	9	7	7	21.80	251.11
freecell-ipc3	3	1	1	13.76	498.80	2	1	1	108.22	90.28	2	1	1	30.44	276.60
grid-ipc1	1	0	0			1	1	1	2.15	220.77	1	0	0		
gripper-ipc1	5	3	3	1.00	169.40	5	3	3	1.06	48.29	5	3	3	1.01	151.66
logistics-ipc2	21	18	18	1.00	205.06	15	22	15	3175.99	14.70	14	21	14	671.66	173.35
logistics-ipc1	3	3	2	24.90	349.40	2	6	2	117.37	19.54	2	3	2	68.76	334.78
miconic-strips-ipc2	45	35	35	1.19	58.03	42	30	30	1.63	94.20	40	30	30	1.38	91.67
mprime-ipc1	17	9	9	31.28	691.51	17	18	17	305.64	67.58	17	13	13	41.99	656.97
mystery-ipc1	16	11	11	2.27	391.64	15	14	14	65.86	37.74	16	13	13	18.47	519.63
openstacks-ipc5	7	5	5	1.23	127.28	7	5	5	2.12	55.96	7	5	5	1.67	269.34
pathways-ipc5	4	4	4	2.10	39.72	4	4	4	1.00	20.14	4	4	4	1.31	42.11
pipes-notank-ipc4	9	1	1	2.88	869.35	11	6	6	8.46	188.93	8	1	1	2.88	1269.40
pipes-tank-ipc4	6	1	1	1.79	1411.60	6	3	3	6.53	120.66	6	1	1	2.44	1440.85
psr-small-ipc4	47	41	41	2.05	103.69	48	46	46	2.58	25.61	47	41	41	1.90	86.39
rovers-ipc5	5	4	4	1.49	46.78	6	4	4	1.00	15.96	6	4	4	1.07	41.53
satellite-ipc4	6	4	4	20.60	75.80	6	5	5	51.06	140.51	5	5	5	54.67	147.09
schedule-strips	42	27	26	38.58	272.33	35	35	34	22.95	97.75	39	9	9	6.94	522.40
tpp-ipc5	5	5	5	7.03	44.16	5	5	5	50.60	10.99	5	5	5	15.84	48.32
trucks-ipc5	5	2	2	1.93	121.48	5	3	3	1.02	111.15	5	2	2	1.02	168.00
zenotravel-ipc3	8	8	7	33.08	191.45	9	10	9	120.20	109.58	8	9	8	12.62	219.12
	294	207	204	9.97	211.16	282	265	248	242.23	61.81	274	196	186	60.47	246.58
				9.55	319.34				186.62	71.54				43.98	354.53

Table 5.1: A summary of the experimental results for the $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$ heuristics. Per heuristic, the first three columns capture the number of tasks solved under the (U) uniform action cost partition, (O) optimal per-search-node action cost partition, and (SB) both these approaches. The fourth and fifth columns capture the difference between the two approaches in terms of expanded nodes and search-node evaluation time.

(i) all the constraints on $\overline{\varphi}$ (given by Eqs. A.160–A.174) are of the form $\overline{\varphi} = \min\{\psi \mid \psi \in \Psi_{\overline{\varphi}}\}$, where $|\Psi_{\overline{\varphi}}| \leq 4$ and each $\psi \in \Psi_{\overline{\varphi}}$ is a linear composition of some action costs, and

(ii) each such constraint can be replaced by maximizing $z_{\overline{\varphi}}$ under the (now linear) constraints $\{z_{\overline{\varphi}} \leq \psi' \mid \psi \in \Psi_{\overline{\varphi}}\}$ where ψ' is obtained from ψ by replacing the action costs in the latter with the respective free variables from $\overrightarrow{w}$.

For instance, for each planning variable v with $\mathsf{pred}(v) = \emptyset$, and each of its goal-valid value-changing sequences $\sigma \in \trianglerighteq^*[\sigma(v)]$, if $\overline{\varphi} = \varphi_v(\sigma)$, then we create the constraint set according to Eq. A.160 as

$$z_{\overline{\varphi}} \leq \left\lceil \frac{|\sigma| - 1}{2} \right\rceil \cdot w_a + \left\lfloor \frac{|\sigma| - 1}{2} \right\rfloor \cdot w_{a'},$$

for each $a, a' \in A_v$ such that $\mathsf{eff}(a)[v] = 1$ and $\mathsf{eff}(a')[v] = 0$. The rest of the constraint sets for $z_{\overline{\varphi}} \in \overrightarrow{z}$ are constructed in the same manner according to Eqs. A.161–A.174. The union of all these constraint sets together with the LP formulation as in Eq. 5.9 results in a linear program satisfying Eq. 5.12.

■

5.5 Experimental Evaluation

While we have proved that the optimal action cost partitioning for abstraction heuristics in use is polynomial-time computable, it is not clear at first glance that it is useful in practice. An almost immediate source of skepticism is that our optimization procedure requires solving a large LP at

every search node, while such per-node computations are typically expected to be of *low* polynomial time. Nonetheless, the superior informativeness of optimal additive heuristics might eventually outweigh the cost of heuristic computation due to the substantial reduction in the number of expanded search nodes. We put this hypothesis to an empirical test and evaluated the practical attractiveness of the optimal fork-decomposition heuristics on a wide sample of planning domains from the International Planning Competitions 1998-2006. The domains were selected to allow a comparison with the results presented by Katz and Domshlak (2009b). We implemented three optimal additive fork-decomposition heuristics within the standard heuristic forward-search framework of the Fast Downward planner (Helmert, 2006), using the A^* algorithm with full duplicate elimination.

- The $h^{\mathcal{F}}$ heuristic corresponds to the ensemble of all (not clearly redundant) fork subgraphs of the causal graph, with the domains of the roots being abstracted using the "leave-one-value-out" binary-valued domain decompositions.

- The $h^{\mathcal{J}}$ heuristic is the same but for the inverted fork subgraphs, with the domains of the roots being abstracted using the "distance-to-goal-value" ternary-valued domain decompositions.

- The ensemble of the $h^{\mathcal{FJ}}$ heuristic is the union of these for $h^{\mathcal{F}}$ and $h^{\mathcal{J}}$.

The linear programs were solved using the interior point method implementation of the MOSEK solver (MOSEK, 2009). The heuristics were compared to the same three additive fork-decomposition heuristics, with action cost partitioning set to "uniform" (Katz & Domshlak, 2009b)[2]. All the experiments were run on a 3 GHz Intel E8400 CPU; the time and memory limits were set to 30 minutes and 1.5 GB, respectively.

The detailed results of the evaluation are relegated to Tables B.15-B.19 in Appendix B and summarized here in Table 5.1. From left to right, each section of the table pertains to $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$, respectively. The first two columns of each section capture the number of planning tasks that were solved using the (U) fixed uniform and (O) optimal per-search-node action cost partitions, respectively. The third column (SB) captures the number of planning tasks that were solved by both these approaches; the last row of these columns provides the respective number of solved planning tasks. The tasks solved under both uniform and optimal action cost partitions are the basis for our comparison of the two approaches in terms of expanded nodes and run time. With respect to these tasks, the fourth and fifth columns respectively capture the (average per domain) *decrease* in the number of expanded nodes and *increase* in the search node evaluation time. The last two rows of these columns average these quantities. The upper summary row depicts the average over all the solved tasks, and the lower row depicts the average of domain averages.

Two conclusions are apparent from Table 5.1. First, switching from the ad hoc fixed to optimal action cost partition almost consistently reduced the number of expanded nodes[3]; the reduction hit three orders of magnitude with the $h^{\mathcal{J}}$ heuristic on the LOGISTICS-IPC2 domain. As expected, however, computing the optimal heuristic estimate is typically between two and three orders of magnitude more time consuming than relying on a fixed action cost partition. At least under the 30 minute time limit per planning task, this payment in per-search-node evaluation time typically

[2]The cost of each action is equally partitioned among all the representatives of that action in the ensemble. For the comparison of the latter with several state-of-the-art heuristics and planners (on the same machines and in the same setting), see Katz and Domshlak (2009b).

[3]On some problems solved under optimal action cost partitioning, our LP solver failed for a few search nodes. To avoid terminating the search, the heuristic value for these nodes was set to 1.

offset the reduction in the number of expanded nodes—the overall number of tasks solved using the optimal additive heuristics was in the end substantially lower. Still, note that numerous tasks that were not solved using the uniform action cost partition were solved using the optimal one; see the LOGISTICS-IPC1 and ZENOTRAVEL domains with all three heuristics, the AIRPORT domain with $h^{\mathcal{J}}$, the BLOCKSWORLD domain with $h^{\mathcal{J}}$ and $h^{\mathcal{FJ}}$, the LOGISTICS-IPC2 domain with $h^{\mathcal{J}}$ and $h^{\mathcal{FJ}}$, the MPRIME domain with $h^{\mathcal{J}}$, and the SCHEDULE-STRIPS domain with $h^{\mathcal{F}}$ and $h^{\mathcal{J}}$.This observation suggests that a compromise should be sought between the accuracy of optimized additive heuristic estimates and the low cost of exploiting a fixed action cost partition. One way of doing so is outlined and evaluated in what follows.

5.6 Beyond Optimal Cost Partitioning

Indeed, using an entirely ad hoc, fixed action cost partition for all search states and optimizing the action cost partition for every individual state are just two extremes of what one can do with a given set of heuristics. A possible middle ground would be to compute a set of action cost partitions that are optimal for *some* states and use them to evaluate *all* the states examined during the search. If that set of action cost partitions is relatively small, then the runtime complexity of heuristic computation can sometimes be further reduced using a *databased* approach in which most of the per node calculations can be shared and precomputed; this technique is inherently natural for explicit abstractions, but it can be very effective with implicit abstractions as well (Katz & Domshlak, 2009b).

To examine the practical relevance of such a middle ground, we took the most conservative evaluation time setup in which at most one heuristic optimization is performed. All the search states were evaluated under the same fixed action cost partition, whose goal was not to be ad hoc but *optimal with respect to the initial state* of the task. Taking our three fork-decomposition heuristic ensembles as a basis for our evaluation, we empirically compared this setup to using a uniform, ad hoc action cost partition. In both cases we used the implicit abstraction database approach of Katz and Domshlak (2009b). Since on some tasks the optimal additive heuristics cannot be computed within a reasonable time even for just a single state, we placed a strict time limit of one minute on the optimization procedure. If the LP solver failed to optimize the initial state action cost partition within that time bound, then the search was executed using the basic uniform action cost partition.

The detailed results of this evaluation are given in Tables B.20-B.25 in Appendix B, and summarized in Table 5.2. From left to right, each section of the table pertains to $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$, respectively. As in Table 5.1, the first two columns of each section of Table 5.2 capture the number of planning tasks that were solved using the optimal for the initial state (O_I) and uniform, ad hoc (U) action cost partitions, respectively. The third column (SB) captures the number of planning tasks that were solved by both these approaches; the last row provides the respective number of solved planning tasks. As there is no real difference in terms of heuristic evaluation time between the uniform and any other fixed action cost partition, the search efforts are compared only in terms of the number of expanded nodes. For each of the three heuristics, the last two columns in its section are devoted to this analysis. The values in those columns depict a measure of informativeness in terms of expanded nodes. The specific measure for comparison is as follows. For each of the two action cost partition schemes, each task contributes a value equal to the minimal number of expanded nodes among the two schemes divided by the number of expanded nodes under the

domain ($\mathcal{D}$)	$h^{\mathcal{F}}$					$h^{\mathcal{J}}$					$h^{\mathcal{FJ}}$				
	O_I	U	SB	E(O_I)	E(U)	O_I	U	SB	E(O_I)	E(U)	O_I	U	SB	E(O_I)	E(U)
airport-ipc4	22	22	22	22.00	21.98	22	20	20	20.00	9.55	21	21	21	21.00	20.17
blocks-ipc2	21	21	21	14.15	18.84	21	18	18	18.00	2.63	21	18	18	17.79	5.07
depots-ipc3	7	7	7	5.14	7.00	7	4	4	4.00	0.91	7	7	7	6.89	5.50
driverlog-ipc3	12	12	12	10.77	7.36	13	12	12	11.10	7.32	12	12	12	11.09	6.23
freecell-ipc3	5	5	5	3.91	5.00	4	4	4	4.00	2.63	5	4	4	3.90	4.00
grid-ipc1	2	2	2	2.00	2.00	2	1	1	1.00	1.00	1	1	1	1.00	1.00
gripper-ipc1	7	7	7	7.00	7.00	7	7	7	7.00	6.90	7	7	7	6.96	6.99
logistics-ipc2	24	22	22	18.37	21.93	21	16	16	16.00	0.92	21	16	16	16.00	1.69
logistics-ipc1	6	6	6	6.00	1.14	5	4	4	4.00	0.87	5	5	5	5.00	0.76
miconic-strips-ipc2	53	51	51	48.79	43.04	53	50	50	49.68	31.94	53	50	50	49.05	34.60
mprime-ipc1	23	23	23	23.00	12.82	23	22	21	20.91	7.50	21	21	21	19.13	12.60
mystery-ipc1	21	21	21	20.90	16.80	18	18	18	18.00	9.81	21	21	21	20.37	17.15
openstacks-ipc5	7	7	7	7.00	5.74	7	7	7	7.00	3.21	7	7	7	7.00	4.09
pathways-ipc5	4	4	4	4.00	2.36	4	4	4	4.00	4.00	4	4	4	4.00	3.23
pipes-notank-ipc4	17	17	17	17.00	12.68	18	15	15	15.00	4.17	16	16	16	16.00	13.30
pipes-tank-ipc4	11	11	11	11.00	7.28	11	9	9	9.00	3.27	9	9	9	9.00	5.11
psr-small-ipc4	49	49	49	36.35	48.87	48	49	48	38.39	47.90	49	49	49	38.89	48.53
rovers-ipc5	7	6	6	5.73	4.06	7	7	7	6.32	5.98	7	6	6	5.48	5.07
satellite-ipc4	6	6	6	5.83	2.97	7	6	6	5.67	2.56	7	6	6	5.70	2.67
schedule-strips	49	46	44	24.83	31.75	49	40	40	30.87	27.14	47	46	46	38.12	40.33
tpp-ipc5	6	6	6	6.00	5.03	6	6	6	6.00	3.56	6	6	6	6.00	4.03
trucks-ipc5	6	6	6	5.08	5.89	7	7	7	7.00	6.20	7	7	7	6.82	6.92
zenotravel-ipc3	13	11	11	10.34	3.73	11	11	11	9.96	6.97	13	11	11	10.48	5.07
	378	368	366	315.18	295.27	371	337	335	312.90	196.94	367	350	350	325.67	254.09

Table 5.2: A summary of the experimental results for the databased $h^{\mathcal{F}}$, $h^{\mathcal{J}}$, and $h^{\mathcal{FJ}}$ heuristics with optimal for the initial state (O_I) and uniform (U) action cost partitions. The last two columns per heuristic depict the measure of success in terms of expanded nodes, with each entry being the sum of our measure over all the tasks in the domain solved under both action cost partitions. The last row in those columns provides the overall measures.

respective scheme. If the denominator is 0 (if, e.g., the initial state is a goal state, or the heuristic estimate of the initial state is ∞), then this value is defined to be 1. As we are interested in comparing expanded nodes, we account only for tasks solved under both action cost partition schemes, and thus the nominator is always well defined. Each task contributes a value of 1 to the winning scheme and a value in [0,1] to the other. For example, if A^* on some task Π with the optimized action cost partition opens 1000 nodes and with the uniform action cost partition it opens 3000 nodes, then Π contributes 1 to the measure E(O_I) and 1/3 to E(U). The last row provides this measure over all the examined domains. Note that in general this measure accounts for all the tasks, giving 0 for unsolved tasks. However, our goal is to compare the expanded nodes, and thus we ignore tasks solved only under one of the formulations. Since the tasks are compared in pairs, the original measure can be obtained from this one by adding the difference between the number of tasks solved under the respective action cost partition scheme and both schemes.

The results depicted in Table 5.2 are very positive. For all three heuristics, for most domains, the number of expanded nodes decreases when moving from uniform cost partitioning to optimal for the initial state. On almost all other domains, the number of expanded nodes increases but not enough to prevent us from solving the same number of tasks as before. There are only four cases in which the increase in node expansions leads to not solving the task, namely $h^{\mathcal{F}}$ on tasks 6-4 and 6-6 from SCHEDULE-STRIPS domain $h^{\mathcal{J}}$ on task 8 from MPRIME domain and task 48 from PSR domain. There is only one domain in which the increase in node expansions leads to fewer tasks being solved, notably $h^{\mathcal{J}}$ on the PSR domain. However, for all three heuristics, the number of tasks solved across the domains increases.

As a final note, we would like to emphasize that more sophisticated setups of partial exploitation of additive heuristics optimization should be even more beneficial. The optimal for the initial state action cost partition should typically lose its attractiveness somewhere along the search, and after a

certain point even the uniform action cost partitioning should be expected to provide more accurate heuristic estimates. Hence, developing meta-reasoning procedures for deciding when (if at all) some effort should be invested in devising additional action cost partitions is clearly of practical interest.

Chapter 6

Summary and Future Work

Our goal in this research was to stratify and extend the machinery of admissible heuristic functions for cost-optimal planning. Our contributions towards this goal are in several complimentary directions. First, we studied the complexity of cost-optimal classical planning with respect to the interplay between the topology of the problem's causal graph and certain types of local structural restrictions of the problems induced by their action sets. For some problem classes, we have shown that cost-optimal planning is tractable, and that relaxing the aforementioned restrictions results in NP-hard problem classes. We believe there is room for further extending the palette of tractable cost-optimal planning, and we urge continuing research in this direction.

Exploiting our tractability results for cost-optimal planning, we then introduced a domain-independent framework for devising admissible heuristics using what we call additive implicit abstractions. Each such implicit abstraction corresponds to abstracting the planning task at hand by an instance of a tractable fragment of optimal planning. The key motivation for our framework was to escape the restriction of explicit abstractions, such as pattern-database and merge-and-shrink abstractions, to abstract spaces of a fixed size. We presented a concrete scheme for additive implicit abstractions by decomposing the planning task along its causal graph and suggested a concrete realization of this idea, called fork-decomposition, that is based on two novel fragments of tractable cost-optimal planning. We then studied the induced admissible heuristics both formally and empirically, and showed that they favorably compete in informativeness with the state-of-the-art admissible heuristics both in theory and in practice. Our empirical evaluation stressed the tradeoff between the accuracy of the heuristics and runtime complexity of computing them. To alleviate the computational overhead of fork-decomposition heuristics, we showed that an equivalent of the explicit abstraction notion of "database" exists also for the fork-decomposition abstractions, despite their exponential-size abstract spaces. Our subsequent empirical evaluation of heuristic search with such databases for the fork-decomposition heuristics showed that it favorably competes with the state of the art of cost-optimal planning.

Another chapter of our work is devoted to additive ensembles of admissible heuristics. Numerous recent works have suggested that additive ensembles of admissible heuristics are a powerful tool for heuristic-search systems. However, the action-cost partition parameter of such ensembles left the "how to add (if at all)" question completely open. We have developed a procedure that closes this question for arbitrary ensembles of all abstraction heuristics with which we are familiar, including pattern databases, constrained pattern databases, merge-and-shrink abstractions, fork-decomposition implicit abstractions, and implicit abstractions based on tractable constraint

optimization over tree-shaped constraint networks. The procedure is based on a linear-programming formulation of the optimization problem: given a classical planning task, a forward-search state, and a set of abstraction-based admissible heuristics, the procedure constructs an optimal additive composition of these heuristics with respect to the search state in question. Most importantly, the time complexity of our procedure is polynomial for arbitrary ensembles of all the above abstraction heuristics.

While more informative estimates are naturally appealing, the impractical evaluation time required to solve large-scale linear problems at each search state forced us to seek more pragmatic alternatives. One such alternative is to calculate the optimal cost partitioning not for *every* state but only for some (small) subset of states, and use some predefined cost partitioning for the rest [1]. We performed an extensive experimental evaluation, calculating the optimal cost partition only in the initial state. In most cases the informativeness of the heuristic increased, allowing us to solve more tasks overall.

The basic principles of the implicit abstraction framework motivate further research in numerous directions, most importantly in

1. discovering new islands of tractability of optimal planning, and
2. abstracting the general planning tasks into such islands.

Likewise, there is promise in combining implicit abstractions with other techniques for deriving admissible heuristic estimates. A first step towards combining implicit abstractions with polynomial-time discoverable landmarks of the planning tasks has recently been taken by Domshlak, Katz, and Lefler (2010). We believe that various combinations of such techniques might well improve the informativeness of the heuristics without substantially increasing their runtime complexity.

Probably the most important entirely open problem is *structure optimization*. While our framework optimizes the composition of a *given* set of TG-structures, our ultimate goal is to move to even more parametric ensembles of this type, allowing flexibility in the actual choice of TG-structures. For instance, it would clearly help to know what PDBs should (optimally) be added to the ensemble, what domain abstractions should (optimally) be performed on the roots of the inverted forks and forks, or what polytrees should (optimally) span the causal graph of the task.

When considering additive composition of admissible heuristics, one question that arises almost immediately is whether a composition based on techniques other than abstraction can be optimized as well. The current answer to this question is affirmative only in part, and this part turns out to be closely related to our results here. Karpas and Domshlak (2009) have recently shown that additive heuristics based on landmarks of the problem's delete-relaxation (such as h^{L} and h^{LA} of Karpas and Domshlak (2009) and $h^{\text{LM-cut}}$ of Helmert and Domshlak (2009)) can be efficiently optimized by solving a certain compact linear program. That both abstraction and landmark additive heuristics optimization procedures are based on linear programming is no coincidence: a recently established connection between such landmark heuristics and the merge-and-shrink abstractions (Helmert & Domshlak, 2009) implies that Karpas and Domshlak's procedure is effectively a *special case* of the LP-optimization procedure for the abstractions developed in our work.

In contrast to abstraction and landmark heuristics, the question of additive ensemble optimization remains open for the h^m family of critical path heuristics (Haslum & Geffner, 2000) even for

[1] Note that the extreme case here would be to give up calculating the optimal cost partitioning altogether for the sake of speed, using some ad hoc (uniform) cost partition for all states, as we did in the databased implicit abstractions case.

the h^1 (also known as $h_{\max}$) member of this family. While computing the additive h^m heuristic for a fixed m is poly-time, this computation is not based on an additive abstraction of the planning task, or, at least, not on a *fixed* abstraction.[2] The state graph over which each h^m is computed is an AND/OR-graph (and not an OR-graph, as is the case for transition graphs), and the actual computation of h^m corresponds to computing a critical tree (and not a shortest path) to the goal. Tangentially, the problem of computing a critical tree in an AND/OR-graph does not appear to have an LP reformulation. Hence, the complexity of computing the optimal additive h^m heuristic is still an open and very interesting question.

Also worth mentioning is that, at first glance, the basic idea of LP-optimizing heuristic composition naturally extends also to *intractable* planning relaxations that admit "second-order" LP-relaxations. For instance, some intractable planning relaxations formalizable via integer-valued LPs (such as the deletes-ignoring relaxation underlying h^+, or the more recent action-ordering relaxation of van den Briel, Benton, Kambhampati, & Vossen, 2007) appear to be quite natural candidates. Things, however, are more complicated than that because, in short, Definition 19 (p. 84) requires a very specific type of LP-encoding. Encodings of this type must satisfy Eq. 5.4 (p. 84), but we know of no ILP-to-LP "second-order" relaxations that meet this requirement. Thus, the most interesting question of incorporating such relaxations still remains open.

Finally, considering the empirical promise of optimal action cost partitioning, we note that the linear programs induced by our LP-encoding technique have a specific structure called in the literature *primal block-angular* (Castro, 2007). This structure can possibly be exploited for devising a more efficient algorithm for optimal action cost partitioning, and this is an interesting venue for future research. Tangentially, the development of automated procedures for devising fixed-size portfolios of action cost partitions is a promising direction, as all the states examined during the search might be "well served" by these procedures.

We believe that pursuing research in all these directions may substantially increase the informativeness and runtime efficiency of implicit abstraction heuristics both *per se* and in the scope of additive ensembles of admissible heuristics.

[2] To the best of our knowledge, the precise relation between critical path and abstraction heuristics is currently an open question, the only exception being additive h^1, now known to be closely related to the landmark heuristics (Helmert & Domshlak, 2009).

Bibliography

Allen, J., Hendler, J., & Tate, A. (Eds.). (1990). *Readings in Planning*. Morgan-Kaufmann.

Bacchus, F., & Yang, Q. (1994). Downward refinement and the efficiency of hierarchical problem solving. *Artificial Intelligence*, *71*(1), 43–100.

Bäckström, C., & Klein, I. (1991). Planning in polynomial time: The SAS-PUBS class. *Computational Intelligence*, *7*(3), 181–197.

Bäckström, C., & Nebel, B. (1995). Complexity results for SAS^+ planning. *Computational Intelligence*, *11*(4), 625–655.

Ball, M., & Holte, R. C. (2008). The compression power of symbolic pattern databases. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 2–11, Sydney, Australia.

Bonet, B., & Geffner, H. (2001). Planning as heuristic search. *Artificial Intelligence*, *129*(1–2), 5–33.

Bonet, B., & Geffner, H. (2006). Heuristics for planning with penalties and rewards using compiled knowledge. In *Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pp. 452–462.

Boutilier, C., Brafman, R., Domshlak, C., Hoos, H., & Poole, D. (2004). CP-nets: A tool for representing and reasoning about conditional *ceteris paribus* preference statements. *Journal of Artificial Intelligence Research*, *21*, 135–191.

Brafman, R. I., & Domshlak, C. (2003). Structure and complexity of planning with unary operators. *Journal of Artificial Intelligence Research*, *18*, 315–349.

Bylander, T. (1994). The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, *69*(1-2), 165–204.

Castro, J. (2007). An interior-point approach for primal block-angular problems. *Computational Optimization and Applications*, *36*(2-3), 195–219.

Chapman, D. (1987). Planning for conjunctive goals. *Artificial Intelligence*, *32*(3), 333–377.

Chen, H., & Giménez, O. (2008). Causal graphs and structurally restricted planning. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 36–43, Sydney, Australia.

Clarke, E., Grumberg, O., & Peled, D. (1999). *Model Checking*. MIT Press.

Coles, A. I., Fox, M., Long, D., & Smith, A. J. (2008). Additive-disjunctive heuristics for optimal planning. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 44–51.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to Algorithms*. MIT Press.

Culberson, J., & Schaeffer, J. (1998). Pattern databases. *Computational Intelligence*, *14*(4), 318–334.

Dean, T., & Wellman, M. (1991). *Planning and Control*. Morgan-Kaufmann.

Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann.

Dechter, R., & Pearl, J. (1985). Generalized best-first search strategies and the optimality of a*. *J. ACM*, *32*(3), 505–536.

Domshlak, C., & Dinitz, Y. (2001). Multi-agent off-line coordination: Structure and complexity. In *Proceedings of Sixth European Conference on Planning (ECP)*, pp. 277–288.

Domshlak, C., & Hoffmann, J. (2006). Fast probabilistic planning through weighted model counting. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 243–252.

Domshlak, C., Hoffmann, J., & Sabharwal, A. (2009). Friends or foes? On planning as satisfiability and abstract CNF encodings. *Journal of Artificial Intelligence Research*, *36*, 415–469.

Domshlak, C., Katz, M., & Lefler, S. (2010). When abstractions met landmarks. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 50–56, Toronto, Canada.

Dräger, K., Finkbeiner, B., & Podelski, A. (2006). Directed model checking with distance-preserving abstractions. In Valmari, A. (Ed.), *Proceedings of the 13th International SPIN Workshop on Model Checking Software*, Vol. 3925 of *Lecture Notes in Computer Science*, pp. 19–36, Berlin Heidelberg. Springer-Verlag.

Edelkamp, S. (2001). Planning with pattern databases. In *Proceedings of the European Conference on Planning (ECP)*, pp. 13–34.

Edelkamp, S. (2002). Symbolic pattern databases in heuristic search planning. In *Proceedings of the International Conference on AI Planning and Scheduling (AIPS)*, pp. 274–293.

Edelkamp, S. (2006). Automated creation of pattern database search heuristics. In *Proceedings of the 4th Workshop on Model Checking and Artificial Intelligence (MoChArt)*.

Edelkamp, S., & Kissmann, P. (2009). Optimal symbolic planning with action costs and preferences. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1690–1695, Pasadena, CA, US.

Erol, K., Nau, D. S., & Subrahmanian, V. S. (1995). Complexity, decidability and undecidability results for domain-independent planning. *Artificial Intelligence, Special Issue on Planning*, *76*(1–2), 75–88.

Felner, A., Korf, R. E., & Hanan, S. (2004). Additive pattern database heuristics. *Journal of Artificial Intelligence Research*, *22*, 279–318.

Fikes, R. E., & Nilsson, N. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *AI Magazine*, *2*, 189–208.

Garey, M. R., & Johnson, D. S. (1978). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New-York.

Geffner, H. (2002). Perspectives on artificial intelligence planning. In *Proceedings of the Eighteenth National Conference on Artificial intelligence (AAAI)*, pp. 1013–1023.

Gerevini, A., Saetti, A., & Serina, I. (2003). Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research*, *20*, 239–290.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated Planning*. Morgan Kaufmann.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, *4*(2), 100–107.

Haslum, P. (2006). *Admissible Heuristics for Automated Planning*. Ph.D. thesis, Linköping University, Department of Computer and Information Science.

Haslum, P. (2008). Additive and reversed relaxed reachability heuristics revisited. In *Proceedings of the 6th International Planning Competition*.

Haslum, P., Bonet, B., & Geffner, H. (2005). New admissible heuristics for domain-independent planning. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI)*, pp. 1163–1168.

Haslum, P., Botea, A., Helmert, M., Bonet, B., & Koenig, S. (2007). Domain-independent construction of pattern database heuristics for cost-optimal planning. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI)*, pp. 1007–1012.

Haslum, P., & Geffner, H. (2000). Admissible heuristics for optimal planning. In *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems (ICAPS)*, pp. 140–149.

Helmert, M. (2003). Complexity results for standard benchmark domains in planning. *Artificial Intelligence*, *146*(2), 219–262.

Helmert, M. (2004). A planning heuristic based on causal graph analysis. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 161–170, Whistler, Canada.

Helmert, M. (2006). The Fast Downward planning system. *Journal of Artificial Intelligence Research*, *26*, 191–246.

Helmert, M., & Domshlak, C. (2009). Landmarks, critical paths and abstractions: What's the difference anyway?. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 162–169, Thessaloniki, Greece.

Helmert, M., Haslum, P., & Hoffmann, J. (2007). Flexible abstraction heuristics for optimal sequential planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 176–183, Providence, RI, USA.

Helmert, M., & Mattmüller, R. (2008). Accuracy of admissible heuristic functions in selected planning domains. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pp. 938–943, Chicago, USA.

Helmert, M. (2008). *Understanding Planning Tasks: Domain Complexity and Heuristic Decomposition*, Vol. 4929 of *Lecture Notes in Computer Science*. Springer.

Hendler, J., Tate, A., & Drummond, M. (1990). Ai planning: systems and techniques. *AI Magazine*, *11*(2), 61–77.

Hernadvölgyi, I., & Holte, R. (1999). PSVN: A vector representation for production systems. Tech. rep. 1999-07, University of Ottawa.

Hoffmann, J. (2003). *Utilizing Problem Structure in Planning: A Local Search Approach*. No. 2854 in LNAI. Springer-Verlag.

Hoffmann, J., & Brafman, R. I. (2006). Conformant planning via heuristic forward search: A new approach. *Artificial Intelligence*, *170*(6-7), 507 – 541.

Hoffmann, J., & Edelkamp, S. (2005). The deterministic part of IPC-4: An overview. *Journal of Artificial Intelligence Research*, *24*, 519–579.

Hoffmann, J., & Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, *14*, 253–302.

Jonsson, A. (2007). The role of macros in tractable planning over causal graphs. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-07)*, pp. 1936–1941.

Jonsson, A., & Giménez, O. (2007). On the hardness of planning problems with simple causal graphs. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 152–159.

Jonsson, P., & Bäckström, C. (1995). Incremental planning. In *New Directions in AI Planning: EWSP'95-3rd European Workshop on Planning*, pp. 79–90, Assisi, Italy.

Jonsson, P., & Bäckström, C. (1998a). State-variable planning under structural restrictions: Algorithms and complexity. *Artificial Intelligence*, *100*(1–2), 125–176.

Jonsson, P., & Bäckström, C. (1998b). Tractable plan existence does not imply tractable plan generation. *Annals of Mathematics and Artificial Intelligence*, *22*(3-4), 281–296.

Karpas, E., & Domshlak, C. (2009). Cost-optimal planning with landmarks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-09)*, pp. 1728–1733, Pasadena, CA, USA.

Katz, M., & Domshlak, C. (2007a). Structural patterns heuristics. In *ICAPS-07 Workshop on Heuristics for Domain-independent Planning: Progress, Ideas, Limitations, Challenges*, Providence, RI, USA.

Katz, M., & Domshlak, C. (2007b). Structural patterns of tractable sequentially-optimal planning. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 200–207, Providence, RI, USA.

Katz, M., & Domshlak, C. (2008a). New islands of tractability of cost-optimal planning. *Journal of Artificial Intelligence Research*, *32*, 203–288.

Katz, M., & Domshlak, C. (2008b). Optimal additive composition of abstraction-based admissible heuristics. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 174–181.

Katz, M., & Domshlak, C. (2008c). Structural patterns heuristics via fork decomposition. In *Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 182–189, Sydney, Australia.

Katz, M., & Domshlak, C. (2009a). Implicit abstraction heuristics.. *submitted*.

Katz, M., & Domshlak, C. (2009b). Structural-pattern databases. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 186–193, Thessaloniki, Greece.

Katz, M., & Domshlak, C. (2010). Optimal admissible composition of abstraction heuristics. *Artificial Intelligence*, *174*, 767–798.

Keyder, E., & Geffner, H. (2008). Heuristics for planning with action costs revisited. In *Proceedings of the 18th European Conference on Artificial Intelligence (ECAI-08)*.

Klein, I., Jonsson, P., & Bäckström, C. (1998). Efficient planning for a miniature assembly line. *Artificial Intelligence in Engineering*, *13*(1), 69–81.

Knoblock, C. (1994). Automatically generating abstractions for planning. *Artificial Intelligence*, *68*(2), 243–302.

Korf, R. E. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, *27*(1), 97–109.

Korf, R. E. (1998). Artificial intelligence search algorithms. In Atallah, M. J. (Ed.), *CRC Handbook of Algorithms and Theory of Computation*, pp. 1–20. CRC Press.

Korf, R. E. (1999). Heuristic search. In Wilson, R. (Ed.), *Encyclopedia of Cognitive Science*, pp. 372–373. MIT Press.

Korf, R. E., & Pearl, J. (1987). Search techniques. In *Annual Review of Computer Science*, Vol. 2, pp. 451–467. Annual Reviews Inc.

McDermott, D., Ghallab, M., Howe, A., Kambhampati, S., Knoblock, C., Ram, A., Veloso, M., Weld, D., & Wilkins, D. (1998). Pddl - the planning domain definition language. Tech. rep., CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control.

Mirkis, V., & Domshlak, C. (2007). Cost-sharing approximations for h^+. In *Proceedings of the 17th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 240–247.

MOSEK (2009). The MOSEK Optimization Tools Version 6.0 (revision 61). [Online]. `http://www.mosek.com`.

Muscettola, N., Nayak, P. P., Pell, B., & Williams, B. C. (1998). Remote Agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, *103*(1-2), 5–47.

Newell, A., & Simon, H. A. (1963). GPS: A program that simulates human thought. In Feigenbaum, E. A., & Feldman, J. (Eds.), *Computers and Thought*, pp. 279–293. Oldenbourg.

Pearl, J. (1984). *Heuristics - Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

Prieditis, A. (1993). Machine discovery of effective admissible heuristics. *Machine Learning*, *12*, 117–141.

Refanidis, I., & Vlahavas, I. P. (2001). The GRT planning system: Backward heuristic construction in forward state-space planning. *Journal of Artificial Intelligence Research*, *15*, 115–161.

Richter, S., Helmert, M., & Westphal, M. (2008). Landmarks revisited. In *Proceedings of the Twenty-Third National Conference on Artificial Intelligence (AAAI)*, pp. 975–982, Chicago, IL, USA.

Russell, S., & Norvig, P. (2004). *Artificial Intelligence: A Modern Approach* (2 edition). Pearson.

Sacerdoti, E. (1974). Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, *5*, 115–135.

Schrijver, A. (1998). *Theory of Linear and Integer Programming*. John Wiley & Sons.

Tenenberg, J. D. (1991). Abstraction in planning. In Allen, J. F., Kautz, H. A., Pelavin, R. N., & Tenenberg, J. D. (Eds.), *Reasoning About Plans*, chap. 4, pp. 213–284. Morgan Kaufmann.

van den Briel, M., Benton, J., Kambhampati, S., & Vossen, T. (2007). An LP-based heuristic for optimal planning. In *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming (CP)*, pp. 651–665, Providence, RI, USA.

Vidal, V. (2004). A lookahead strategy for heuristic search planning. In *Proceedings of the 14th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 150–160.

Weld, D. S. (1999). Recent advances in AI planning. *AI Magazine*, *20*(2), 93–123.

Wilkins, D. E. (1984). Domain-independent planning: Representation and plan generation. *Artificial Intelligence*, *22*, 269–301.

Williams, B., & Nayak, P. (1996). A model-based approach to reactive self-configuring systems. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pp. 971–977, Portland, OR. AAAI Press.

Williams, B., & Nayak, P. (1997). A reactive planner for a model-based executive. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1178–1185, Nagoya, Japan.

Yang, F., Culberson, J., & Holte, R. (2007). A general additive search abstraction. Tech. rep. TR07-06, University of Alberta.

Yang, F., Culberson, J., Holte, R., Zahavi, U., & Felner, A. (2008). A general theory of additive state space abstractions. *Journal of Artificial Intelligence Research*, *32*, 631–662.

Zhou, R., & Hansen, E. (2006). Breadth-first heuristic search. *Artificial Intelligence*, *170*, 385–408.

Appendix A

Compexity Results in Detail

A.1 Cost-Optimal Planning for $\mathbf{P}_b$

This section is devoted to the proof of tractability of cost-optimal planning for the problem fragment $\mathbf{P}_b$. We begin with describing our planning-to-COP scheme for $\mathbf{P}_b$, and then prove its correctness and complexity.

A.1.1 Construction

Before we proceed with the details of the construction, here we make an assumption that the specification of our actions is fully specified in terms of the variable parents in the causal graph. If $\mathsf{pred}(v) \subset V$ denotes the set of all the immediate predecessors of v in the causal graph $CG(\Pi)$, then we assume that, for each action $a \in A_v$, $\mathsf{pre}(a)[w]$ is specified for each $w \in \mathsf{pred}(v)$. While in general such an assumption requires an exponential translation, this is not the case with $\mathbf{P}_b$. Let $A^{\bowtie}$ be such a translation of the original problem actions A. To obtain $A^{\bowtie}$, for every variable $v \in V$, every action in A_v is represented in $A^{\bowtie}$ by a set of actions that are preconditioned by complete assignments to $\mathsf{pred}(v)$. If $|\mathsf{pred}(v)| = k$, and the precondition of a is specified only in terms of some $0 \leq k' \leq k$ parents of v, then a is represented in $A^{\bowtie}$ by a set of actions, each extending the precondition $\mathsf{pre}(a)$ by a certain instantiation of the previously unspecified $k - k'$ parents of v, and having the cost $cost^{\bowtie}(a') = cost(a)$. Note that the expansions of two or more original actions may overlap, and thus $A^{\bowtie}$ may contain syntactically identical yet differently priced actions. Without loss of generality, we assume that only a minimally-priced such clone is kept in $A^{\bowtie}$. The key point is that that compiling A into $A^{\bowtie}$ for the $\mathbf{P}_b$ problems is poly-time, as the procedure is linear in $|A^{\bowtie}| = O(n2^{\mathsf{ln}(\Pi)+1})$. Finally, the (straightforward to prove) Proposition 1 summarizes the correctness of our assumption with respect to the cost-optimal planning for UB.

Proposition 1 *For any* UB *task* $\Pi = \langle V, A, I, G, cost\rangle$*, the cost of the optimal plans for* Π *is equal to this for* $\Pi^{\bowtie} = \langle V, A^{\bowtie}, I, G, cost^{\bowtie}\rangle$*, with optimal plans for* Π *being reconstructible in linear time from the optimal plans for* $\Pi^{\bowtie}$ *and vice versa.*

We now specify our compilation of a given $\mathbf{P}_b$ task Π into a constraint optimization problem COP_{Π}. The COP variable set $\mathcal{X}$ contains a variable x_v for each planning variable $v \in V$, and the

domain $\mathcal{D}(x_v)$ consists of all valid prefixes of $\tau(v)$. That is,

$$\begin{aligned} \mathcal{X} &= \{x_v \mid v \in V\} \\ \mathcal{D}(x_v) &= \trianglerighteq^*[\tau(v)] \end{aligned} \tag{A.1}$$

Informally, the domain of each variable x_v contains all possible sequences of values that the planning variable v may undergo along a cost-optimal plan. Now, for each planning variable v with parents $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$, the set of COP functions $\mathcal{F}$ contains a single non-negative, real-valued function φ_v with the scope

$$Q_v = \{x_v, x_{w_1}, \ldots, x_{w_k}\} \tag{A.2}$$

The purpose of these functions is to connect between the value-changing sequences of v and these of its parents $\mathsf{pred}(v)$. The specification of these functions is the more involved part of the compilation.

First, for each planning variable v with $\mathsf{pred}(v) = \emptyset$, and each of its goal-valid (time-stamped) value-changing sequences $\tau' \in \trianglerighteq^*[\tau(v)]$, we set

$$\varphi_v(\tau') = \left\lfloor \frac{|\tau'|}{2} \right\rfloor \cdot C(a_{\mathsf{w}_v}) + \left\lfloor \frac{|\tau'| - 1}{2} \right\rfloor \cdot C(a_{\mathsf{b}_v}) \tag{A.3}$$

where $\mathsf{eff}(a_{\mathsf{w}_v})[v] = \{\mathsf{w}_v\}$, $\mathsf{eff}(a_{\mathsf{b}_v})[v] = \mathsf{b}_v$, and $C(a) = \mathit{cost}(a)$ if $a \in A$, and ∞, otherwise. It is not hard to verify that $\varphi_v(\tau')$ corresponds to the optimal cost of performing $|\sigma'| - 1$ value changes of v in Π.

Now, for each non-root variable v with $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}, k \geq 1$, we specify the function φ_v as follows. For each goal-valid value-changing sequence $\tau' \in \trianglerighteq^*[\tau(v)]$ of v, and each set of such goal-valid value-changing sequences $\{\tau'_1 \in \trianglerighteq^*[\tau(w_1)], \ldots, \tau'_k \in \trianglerighteq^*[\tau(w_k)]\}$ of v's parents, we want to set $\varphi_v(\tau', \tau'_1, \ldots, \tau'_k)$ to the *optimal cost of performing* $|\tau'| - 1$ *value changes of* v*, given that* $w_1, \ldots, w_k$ *change their values* $|\tau'_1| - 1, \ldots, |\tau'_k| - 1$ *times*, respectively. In what follows, we reduce setting the value $\varphi_v(\tau', \tau'_1, \ldots, \tau'_k)$ to solving a single-source shortest path problem on an edge-weighted digraph $G'_e(v)$ that slightly enhances a similarly-named graphical structure suggested by Brafman and Domshlak (2003). Though the construction of $G'_e(v)$ is very similar to this of Brafman and Domshlak (2003), here we provide it in full details to save the reader patching the essential differences from the construction of Brafman and Domshlak (2003).

Given the value-changing sequences $\tau'_1, \ldots, \tau'_k$ as above, the digraph $G'_e(v)$ is created in three steps. First, we construct a labeled directed graph $G(v)$ capturing information about all sequences of assignments on $\mathsf{pred}(v)$ that can enable n or less value flips of v. The graph $G(v)$ is defined as follows:

1. $G(v)$ consist of $\eta = \max_{\tau' \in \trianglerighteq^*[\tau(v)]} |\tau'|$ nodes.

2. $G(v)$ forms a *2-colored multichain*, i.e., (i) the nodes of the graph are colored with black (b) and white (w), starting with black; (ii) there are no two subsequent nodes with the same color; (iii) for $1 \leq i \leq \eta - 1$, edges from the node i are only to the node $i + 1$.

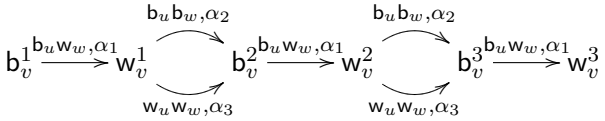
   Observe that such a construction of $G(v)$ promises that the color of the last node will be consistent with the goal value $G[v]$ if such is specified.

3. The nodes of $G(v)$ are denoted precisely by the elements of the longest goal-valid value-changing sequence $\tau' \in \trianglerighteq^*[\tau(v)]$, that is, b^i_v stands for the ith black node in $G(v)$.

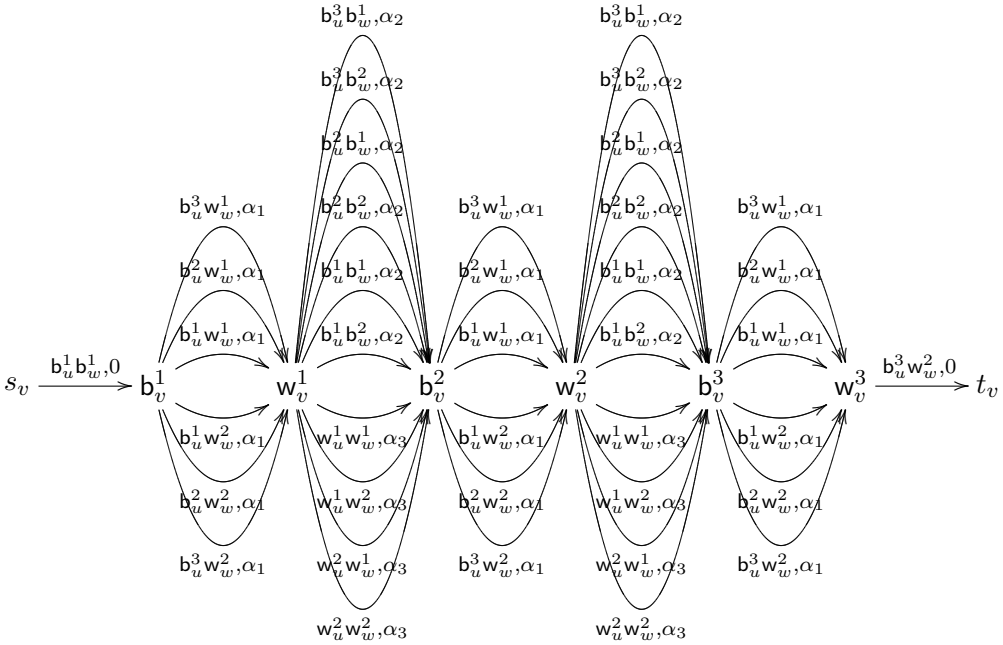
4. Suppose that there are m operators in A_v that, under different preconditions, change the value of v from b_v to w_v. In this case, for each i, there are m edges from b_v^i to w_v^i, and $|A_v|-m$ edges from w_v^i to b_v^{i+1}. Each such edge e is labeled with the cost of the corresponding action, as well as with the prevail conditions of that action, which is a k-tuple of the values of $w_1, \ldots, w_k$. This compound label of e is denoted by $l(e)$, and the prevail condition and cost parts of $l(e)$ are henceforth denoted by $\mathsf{prv}(e)$ and $cost(e)$, respectively.

As the formal definition of $G(v)$ is somewhat complicated, here we provide an illustrating example. Suppose that we are given a $\mathbf{P}_b$ task over 5 variables, and we consider a variable v with $\mathsf{pred}(v) = \{u, w\}$, $I[v] = \mathsf{b}_v$, and $G[v] = \mathsf{w}_v$. Let

$$A_v = \begin{cases} a_1 \ : \ \mathsf{pre}(a_1) = \{\mathsf{b}_v, \mathsf{b}_u, \mathsf{w}_w\}, \ \mathsf{eff}(a_1) = \{\mathsf{w}_v\}, \ cost(a_1) = \alpha_1 \\ a_2 \ : \ \mathsf{pre}(a_2) = \{\mathsf{w}_v, \mathsf{b}_u, \mathsf{b}_w\}, \ \mathsf{eff}(a_2) = \{\mathsf{b}_v\}, \ cost(a_2) = \alpha_2 \\ a_3 \ : \ \mathsf{pre}(a_3) = \{\mathsf{w}_v, \mathsf{w}_u, \mathsf{w}_w\}, \ \mathsf{eff}(a_3) = \{\mathsf{b}_v\}, \ cost(a_3) = \alpha_3 \end{cases}$$



(a)



(b)

Figure A.1: Example of the graphs (a) $G(v)$, and (b) $G'(v)$.

The corresponding graph $G(v)$ is depicted in Figure A.1a. Informally, the graph $G(v)$ captures information about all *potentially possible* executions of the actions in A_v along a cost-optimal plan

for Π. Each path from the source node of $G(v)$ uniquely corresponds to one such an execution. Although the number of these alternative executions may be exponential in n, their graphical representation via $G(v)$ is compact—the number of edges in $G(v)$ is $O(n \cdot |A_v|)$. Note that the information about the number of times each action in A_v can be executed is not captured by $G(v)$. The following two steps add this essential information into the graphical structure.

At the second step, the digraph $G(v) = (V, E)$ is expanded into a digraph $G'(v) = (V', E')$ by substituting each edge $e \in E$ with a set of edges (between the same nodes), but with the labels corresponding to all possible assignments of the elements of $\tau'_1, \ldots, \tau'_k$ to $\mathsf{prv}(e)$. For example, an edge $e \in E$ labeled with $\|\mathsf{b}_{w_1}\mathsf{b}_{w_2}, 10\|$ might be substituted in E' with edges labeled with $\{\|\mathsf{b}^1_{w_1}\mathsf{b}^1_{w_2}, 10\|, \|\mathsf{b}^1_{w_1}\mathsf{b}^2_{w_2}, 10\|, \|\mathsf{b}^2_{w_1}\mathsf{b}^1_{w_2}, 10\|, \ldots\}$. Finally, we set $V' = V \cup \{s_v, t_v\}$, and add a single edge labeled with the first elements of $\tau'_1, \ldots, \tau'_k$ and zero cost (that is, $\|\mathsf{b}^1_{w_1} \cdots \mathsf{b}^1_{w_k}, 0\|$) from s_v to the original source node b^1_v, plus a single edge labeled with the last elements of $\tau'_1, \ldots, \tau'_k$ and zero cost from the original sink node of $G(v)$ to t_v. Informally, the digraph $G'(v)$ can be viewed as a projection of the value-changing sequences $\tau'_1, \ldots, \tau'_k$ on the base digraph $G(v)$. Figure A.1b illustrates $G'(v)$ for the example above, assuming $\tau'_u = \mathsf{b}^1_u \cdot \mathsf{w}^1_u \cdot \mathsf{b}^2_u \cdot \mathsf{w}^2_u \cdot \mathsf{b}^3_u$ and $\tau_w = \mathsf{b}^1_w \cdot \mathsf{w}^1_w \cdot \mathsf{b}^2_w \cdot \mathsf{w}^2_w$.

At the third step, a digraph $G'_e(v) = (V'_e, E'_e)$ is constructed from $G'(v)$ as follows.

(i) The nodes V'_e correspond to the *edges* of $G'(v)$.

(ii) The edges $(v_e, v_{e'}) \in E'_e$ correspond to all pairs of immediately consecutive edges $e, e' \in E'$ such that, for $1 \leq i \leq k$, either $\mathsf{prv}(e)[w_i] = \mathsf{prv}(e')[w_i]$, or $\mathsf{prv}(e')[w_i]$ appears after $\mathsf{prv}(e)[w_i]$ along τ'_i.

(iii) Each edge $(v_e, v_{e'}) \in E'_e$ is weighted with $cost(e')$.

Figure A.2 depicts the graph $G'_e(v)$ for our example.

Assuming $\alpha_3 \leq \alpha_2$, the dashed edges correspond to the *minimal-cost path of length 5* from the dummy source node $\mathsf{b}^1_u\mathsf{b}^1_w$. Note that, if the costs of actions A_v is all we care about, this path corresponds to the cost-optimal sequence of 5 value changes of v starting from its initial value b_v in Π. In fact, not only this path corresponds to such a cost-optimal sequence, but it also explicitly describes the underlying sequence of actions from A_v, as well as the total cost of these actions. Finally, for all $0 \leq i \leq n$, such minimal-cost paths of length i can be determined by running on $G'_e(v)$ a low-polynomial single-source shortest paths algorithm of Dijkstra (Cormen, Leiserson, & Rivest, 1990). This property of the graph $G'_e(v)$ provides us with the last building block for our algorithm for cost-optimal planning for $\mathbf{P}_b$.

The overall algorithm for cost-optimal planning for $\mathbf{P}_b$ that is based on the above construction is depicted in Figure A.3. Given a task $\Pi \in \mathbf{P}_b$, the algorithm compiles it into the constraint optimization problem COP_Π, and solves it using a standard algorithm for constraint optimization over tree constraint networks (Dechter, 2003). The specification of COP_Π has already been explained inline. We believe it is already intuitive that this compilation takes time polynomial in the description size of Π, but in the next section we also prove it formally. Solving COP_Π using the algorithm for tree-structured constraint networks can be done in time polynomial in the description size of COP_Π because

(i) the tree-width of the cost network of COP_Π is bounded by the same constant that bounds the in-degree of the causal graph, and

(ii) optimal tree-decomposition of the COP_Π's cost network is given by any topological ordering of the causal graph.

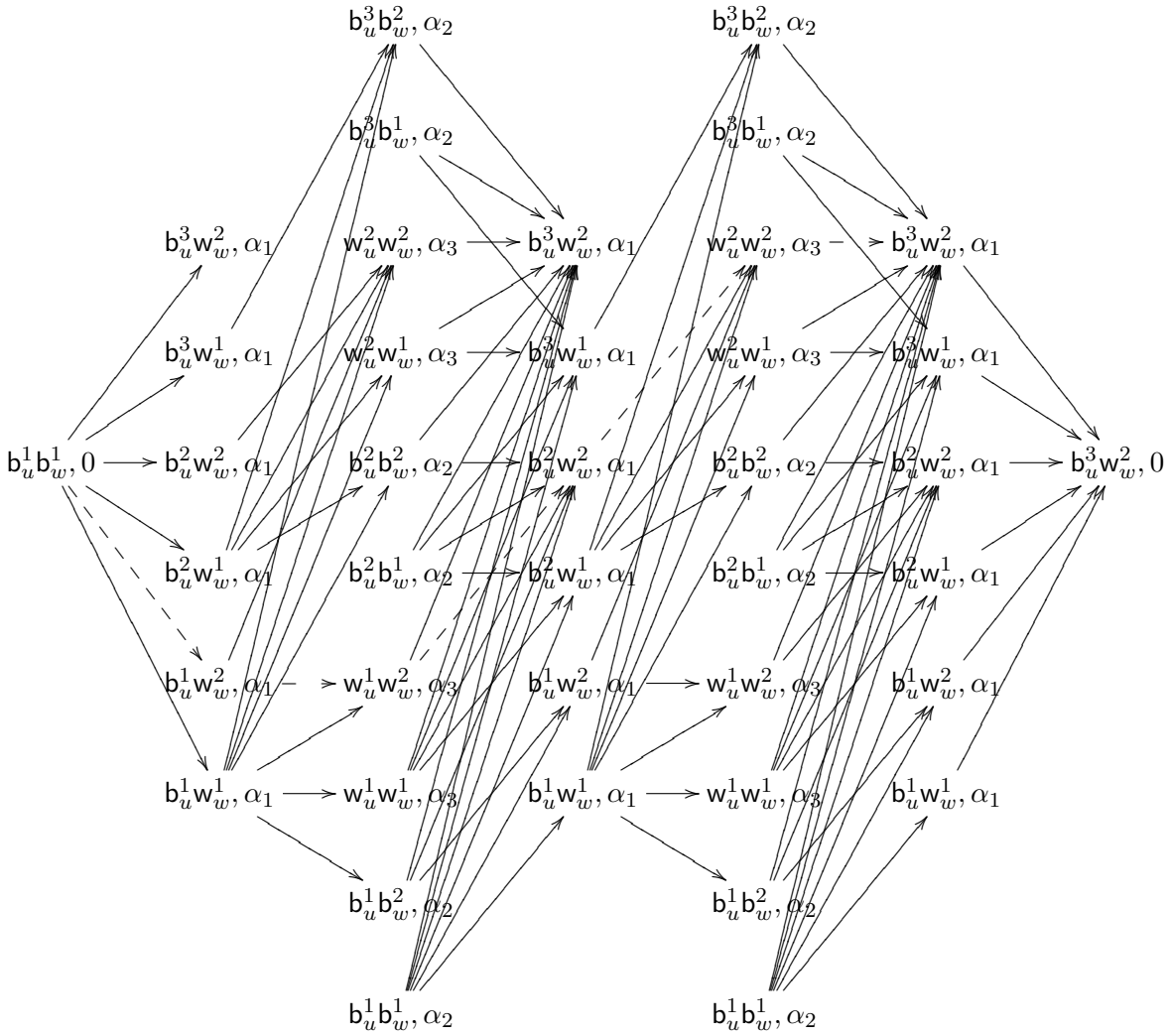


Figure A.2: The graph $G'_e(v)$ constructed from the graph $G'(v)$ in Fugure A.1b.

A.1.2 Correctness and Complexity

We now proceed with proving both the correctness and the polynomial time complexity of our algorithm for $\mathbf{P}_b$. We begin with proving in Theorem 22 below a rather general property of polytrees that helps both here and in our constructs for the $\mathbf{P}(1)$ fragment in the next two sections. We note that a special case of this property has been already exploited in the past in the proof of Lemma 2 of Brafman and Domshlak (2003), but, to our knowledge, this property has never been formulated as a generic claim of Theorem 22. Throughout this chapter we then demonstrate that this generic claim can be helpful in numerous situations.

Theorem 22 *Let G be a polytree over vertices $V = \{1, \ldots, n\}$, and $\mathsf{pred}(i) \subset V$ denote the immediate predecessors of i in G. For each $i \in V$, let O_i be a finite set of objects associated with the vertex i, with the sets $O_1, \ldots, O_n$ being pairwise disjoint. For each $i \in V$, let $>_i$ be a strict partial order over O_i, and, for each $j \in \mathsf{pred}(i)$, let $>_{i,j}$ be a strict partial order over $O_i \cup O_j$.*

If, for each $i \in V, j \in \mathsf{pred}(i)$, the transitively closed $>_i \cup >_{i,j}$ and $>_j \cup >_{i,j}$ induce (strict) partial orders over $O_i \cup O_j$, then so does the transitively closed

$$> \;=\; \bigcup_{i \in V} \left(>_i \cup \bigcup_{j \in \mathsf{pred}(i)} >_{i,j} \right)$$

over $O = \bigcup_{i \in V} O_i$.

Proof: In what follows, by o_i we denote an arbitrary object from O_i. Assume to the contrary that both $>_i \cup >_{i,j}$ and $>_j \cup >_{i,j}$ are (strict) partial orders, and yet $>$ is not so. That is, there exists a pair of objects $o_i, o_j \in O$ for which hold both $o_i > o_j$ and $o_j > o_i$. By the construction of $>$, we have that there is a, possibly empty, path between the vertices i and j in the undirected graph induced by G. Since G is a polytree, we know that such an undirected path between i and j is unique. Thus, we must have

$$\begin{array}{l} \alpha \;:\; o_i = o^1_{i_0} < \ldots < o^{x_0}_{i_0} < o^1_{i_1} < \ldots < o^{x_1}_{i_1} < \ldots\ldots < o^1_{i_m} < \ldots < o^{x_m}_{i_m} = o_j \\ \beta \;:\; o_i = \bar{o}^1_{i_0} > \ldots > \bar{o}^{y_0}_{i_0} > \bar{o}^1_{i_1} > \ldots > \bar{o}^{y_1}_{i_1} > \ldots\ldots > \bar{o}^1_{i_m} > \ldots > \bar{o}^{y_m}_{i_m} = o_j \end{array} \tag{A.4}$$

such that, for $0 \leq k \leq m$, both $x_k \geq 1$ and $y_k \geq 1$, and each step in both chains α and β is *directly implied* by some "local" relation $>_l$ or $>_{l,l'}$ constructing $>$. The corresponding unique undirected path between i and j is:

$$i = i_0 \cdot i_1 \cdot \ldots \cdot i_{m-1} \cdot i_m = j \tag{A.5}$$

Without loss of generality, we assume that the cycle in $>$ induced by α and β is *length-wise minimal* among all such cycles in $>$. In particular, this implies that

(i) for $0 \leq k \leq m$, we have $1 \leq x_k, y_k \leq 2$,

(ii) for each pair of objects $o \in \alpha$, $o' \in \beta$, we have $o \neq o'$, unless $o = o' = o_i$ or $o = o' = o_j$, and

(iii) for each pair of objects $o \in \alpha$, $o' \in \beta$, no $>_l$ (and no $>_{l,l'}$) implies $o' \succ_l o$ (respectively, $o' \succ_{l,l'} o$).

First, let us show that at least one of the chains α and β contains at least one internal element. Assume, to the contrary, that both α and β contain no internal elements. If $i = j$, then we have $o_i >_i o'_i$ (where $o'_i = o_j$) and $o'_i >_i o_i$, contradicting our assumption that $>_i$ is a partial order. (If $>_i$ is not a partial order, then so is each $>_i \cup >_{i,j}$.) Otherwise, if $i \neq j$, then either $i \in \mathsf{pred}(j)$ or $j \in \mathsf{pred}(i)$. Assuming the latter, $(o_i > o_j) \wedge (o_i > o_j)$ implies $(o_i >_{i,j} o_j) \wedge (o_i >_{i,j} o_j)$, contradicting our assumption that $>_{i,j}$ is a partial order.

Given that, let us now prove that $o^{x_m}_{i_m} \neq \bar{o}^{y_m}_{i_m}$, contradicting the assumption that the chains α and β as in Eq. A.4 exist. We do it on a case-by-case basis of possible combinations of x_m, y_m, and length-minimality of the cycle $\alpha\beta$ implies that there are only four such cases to consider.

[$x_m = 2, y_m = 2$] In this case, Eq. A.4 implies $\bar{o}^1_{i_m} >_{i_m} \bar{o}^2_{i_m} = o^2_{i_m} >_{i_m} o^1_{i_m}$. The transitivity of $>_{i_m}$ then implies $\bar{o}^1_{i_m} > o^1_{i_m}$, contradicting our assumption of minimality of the cycle $\alpha\beta$.

[$x_m = 1, y_m = 1$] From Eq. A.5 we have either $i_{m-1} \in \mathsf{pred}(i_m)$ or $i_m \in \mathsf{pred}(i_{m-1})$. If $i_{m-1} \in \mathsf{pred}(i_m)$, then Eq. A.4 implies $\bar{o}^{y_{m-1}}_{i_{m-1}} >_{i_m, i_{m-1}} \bar{o}^1_{i_m} = o^1_{i_m} >_{i_m, i_{m-1}} o^{x_{m-1}}_{i_{m-1}}$. The transitivity of

$>_{i_m,i_{m-1}}$ then implies $\bar{o}_{i_{m-1}}^{y_{m-1}} > o_{i_{m-1}}^{x_{m-1}}$, contradicting our assumption of minimality of the cycle $\alpha\beta$. Otherwise, if $i_m \in \mathsf{pred}(i_{m-1})$, then Eq. A.4 implies $\bar{o}_{i_{m-1}}^{y_{m-1}} >_{i_{m-1},i_m} \bar{o}_{i_m}^1 = o_{i_m}^1 >_{i_{m-1},i_m} o_{i_{m-1}}^{x_{m-1}}$. Again, the transitivity of $>_{i_{m-1},i_m}$ then implies $\bar{o}_{i_{m-1}}^{y_{m-1}} > o_{i_{m-1}}^{x_{m-1}}$, contradicting our assumption of minimality of the cycle $\alpha\beta$.

[$x_m = 2, y_m = 1$] Here as well, Eq. A.5 implies that we have either $i_{m-1} \in \mathsf{pred}(i_m)$ or $i_m \in \mathsf{pred}(i_{m-1})$. If $i_{m-1} \in \mathsf{pred}(i_m)$, then Eq. A.4 implies $\bar{o}_{i_{m-1}}^{y_{m-1}} >_{i_m,i_{m-1}} \bar{o}_{i_m}^1 = o_{i_m}^2 >_{i_m} o_{i_m}^1$. Then, the transitivity of $>_{i_m} \cup >_{i_m,i_{m-1}}$ implies $\bar{o}_{i_{m-1}}^{y_{m-1}} > o_{i_m}^1$, contradicting our assumption of minimality of the cycle $\alpha\beta$. Otherwise, if $i_m \in \mathsf{pred}(i_{m-1})$, then Eq. A.4 implies $\bar{o}_{i_{m-1}}^{y_{m-1}} >_{i_{m-1},i_m} \bar{o}_{i_m}^1 = o_{i_m}^2 >_{i_m} o_{i_m}^1$. Then, the transitivity of $>_{i_m} \cup >_{i_{m-1},i_m}$ implies $\bar{o}_{i_{m-1}}^{y_{m-1}} > o_{i_m}^1$, contradicting our assumption of minimality of the cycle $\alpha\beta$.

[$x_m = 1, y_m = 2$] This case is similar to the previous case of $x_m = 2, y_m = 1$, *mutatis mutandis*.

■

Lemma 9 *Let Π be a planning task in $\mathbf{P}_b$, $\mathsf{COP}_\Pi = (\mathcal{X}, \mathcal{F})$ be the corresponding constraint optimization problem, and $\overline{x} \in \mathcal{D}(\mathcal{X})$ be an optimal solution for COP_Π with $\sum_{\varphi\in\mathcal{F}} \varphi(\overline{x}) = \alpha$.*

(I) If $\alpha < \infty$, then a plan of cost α for Π can be reconstructed from $\overline{x}$ in time polynomial in the description size of Π.

(II) If Π has a plan, then $\alpha < \infty$.

Proof:
(I) Given a COP solution $\overline{x} = \{\tau_{v_1}, \ldots, \tau_{v_n}\}$ with $\sum_{\varphi\in\mathcal{F}} \varphi(\overline{x}) = \alpha < \infty$, we construct a plan ρ for Π with $cost(\rho) = \alpha$.

First, for each variable $v \in V$ with $\mathsf{pred}(v) = \emptyset$, let a sequence ρ_v of actions from A_v be defined as

$$\rho_v = \begin{cases} \emptyset & |\tau_v| = 1 \\ a_v^1 \cdot \ldots \cdot a_v^{|\tau_v|-1} & \text{otherwise} \end{cases}, \tag{A.6}$$

where, for $1 \leq j \leq |\tau_v| - 1$,

$$a_v^j = \begin{cases} a_{\mathsf{b}_v}, & j \text{ is even} \\ a_{\mathsf{w}_v}, & j \text{ is odd} \end{cases}, \tag{A.7}$$

with $\mathsf{eff}(a_{\mathsf{b}_v}) = \{\mathsf{b}_v\}$, and $\mathsf{eff}(a_{\mathsf{w}_v}) = \{\mathsf{w}_v\}$. From Eq. A.3 and $\varphi_v(\tau_v) < \infty$, we immediately have (i) $\{a_{\mathsf{w}_v}\} \subseteq A_v$ if $|\tau_v| \geq 2$, and $\{a_{\mathsf{b}_v}, a_{\mathsf{w}_v}\} \subseteq A_v$ if $|\tau_v| > 2$, and (ii) $cost(\rho_v) = \varphi_v(\tau_v)$. Now, for a purpose that gets clear below, let a binary relation $>_v$ over the action elements of ρ_v be defined as the transitive closure of $\{a_v^{j-1} < a_v^j \mid 1 < j \leq |\tau_v| - 1\}$. Clearly, $>_v$ constitutes a strict total ordering over the elements of ρ_v.

Next, for each non-root variable $v \in V$ with $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$, we construct the graph $G'_e(v)$ with respect to $\tau_{w_1}, \ldots, \tau_{w_k}$, and determine in $G'_e(v)$ a minimal-cost path of $|\tau_v|-1$ edges from the source node $\langle \mathsf{b}_{w_1}^1 \cdots \mathsf{b}_{w_k}^1 \rangle$. The existence of such a path is implied by $\varphi_v(\tau_v, \tau_{w_1}, \ldots, \tau_{w_k}) < \infty$. By the construction of $G'_e(v)$ we also know that, for $1 \leq j \leq |\tau_v| - 1$, the j-th edge on this path is from a node labeled with $\langle \tau_{w_1}[l_1^{j-1}] \cdots \tau_{w_k}[l_k^{j-1}] \rangle$ to a node labeled with $\langle \tau_{w_1}[l_1^j] \cdots \tau_{w_k}[l_k^j] \rangle$, where

for $1 \leq l \leq k$, we have $l_i^0 = 1$ and $l_i^{j-1} \leq l_i^j$. Having that, let a sequence ρ_v of actions from A_v be defined as in Eq. A.6, with, for $1 \leq j \leq |\tau_v| - 1$,

$$\begin{aligned} \mathsf{eff}(a_v^j) &= \{\tau_v[j+1]\} \\ \mathsf{pre}(a_v^j) &= \left\{\tau_v[j], \tau_{w_1}[l_1^j], \tau_{w_2}[l_2^j], \ldots, \tau_{w_k}[l_k^j]\right\} \end{aligned} \tag{A.8}$$

Note that $\{a_v^1, \ldots, a_v^{|\tau_v|-1}\} \subseteq A_v$ is implied by the construction of $G'_e(v)$ and the presence of the considered minimal-cost path in it.

Now, similarly to the case of the root variables, let a binary relation $>_v$ over the action elements of ρ_v be defined as the transitive closure of $\{a_v^{j-1} < a_v^j \mid 1 < j \leq |\tau_v| - 1\}$. Here as well, $>_v$ constitutes a strict total ordering over the elements of ρ_v. In addition, for each parent w_i of v, let a binary relation $>_{v,w_i}$ over the union of the action elements of ρ_v and ρ_{w_i} be defined as the transitive closure of $>_{v,w_i}^- \cup >_{v,w_i}^+$, which in turn are defined as

$$\begin{aligned} >_{v,w_i}^- &= \left\{a_{w_i}^{l_i^j - 1} < a_v^j \mid 1 \leq j \leq |\tau_v| - 1, l_i^j > 1\right\} \\ >_{v,w_i}^+ &= \left\{a_v^j < a_{w_i}^{l_i^j} \mid 1 \leq j \leq |\tau_v| - 1, l_i^j < |\tau_{w_i}|\right\}. \end{aligned} \tag{A.9}$$

It is not hard to verify from Eq. A.9 that, for each $v \in V$ and each $w \in \mathsf{pred}(v)$, not only $>_{v,w}$ constitutes a strict partial ordering, but so are the transitively closed $>_v \cup >_{v,w}$ and $>_w \cup >_{v,w}$. Given that,

- By the definition of $\rho_w = \langle a_w^1 \cdot \ldots \cdot a_w^l \rangle$, and the polytree structure of the causal graph $CG(\Pi)$, restricting the preconditions and effects of each a_w^i to the variables $\{v\} \cup \mathsf{pred}(v)$, we have $\mathsf{pre}(a_w^i) = \{\mathsf{b}_w\}$, $\mathsf{eff}(a_w^i) = \{\mathsf{w}_w\}$ for i being odd, and $\mathsf{pre}(a_w^i) = \{\mathsf{w}_w\}$, $\mathsf{eff}(a_w^i) = \{\mathsf{b}_w\}$ for i being even. For each $1 \leq i \leq k$, from Eq. A.8 we have $\mathsf{eff}(a_{w_i}^{l_i^j - 1}) \in \mathsf{pre}(a_v^j)$. From Eq. A.9 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, the construction of the graph $G'_e(v)$ implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

- The polytree structure of the causal graph $CG(\Pi)$ and Theorem 22 together imply that the transitively closed relation

  $$> = \bigcup_{v \in V} (>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w})$$

  is a strict partial order over the union of the action elements of $\rho_{v_1}, \ldots, \rho_{v_n}$.

Putting thing together, the above implies that any linearization of $>$ constitutes a valid plan ρ for Π with cost

$$cost(\rho) = \sum_{v \in V} cost(\rho_v) = \sum_{v \in V} \varphi_v(\overline{x}),$$

which is exactly what we had to prove. Here we also note that the plan extraction step of the algorithm polytree-k-indegree corresponds exactly to the above construction along Eqs. A.6-A.9, providing us in polynomial time with a concrete cost-optimal plan corresponding to the optimal solution for COP_Π.

(II) We now prove that if Π is solvable, then we must have $\alpha < \infty$. Assume to the contrary that this is not the case. Let Π be a solvable planning task, and let ρ be an irreducible plan for Π. Given such ρ, let $\overline{x}_\rho = \{\tau_{v_1}, \ldots, \tau_{v_n}\}$ be an COP assignment with each $|\tau_{v_i}| = |\rho{\downarrow_{v_i}}| - 1$. Note that $\overline{x}_\rho$ is well-defined (that is, for $1 \leq i \leq n$, we have $\tau_{v_i} \in \trianglerighteq^*[\tau(v_i)]$) by the definition of $\tau(v_i)$, Corollary 1, and ρ being irreducible. Let us now show that $\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}_\rho) \leq cost(\rho)$, contradicting our assumption that $\alpha = \infty$ due to $\alpha \leq \sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}_\rho)$ and $cost(\rho) < \infty$.

First, for each variable v with $\mathsf{pred}(v) = \emptyset$, Eq. A.3 immediately implies $\varphi_v(\overline{x}_\rho) \leq cost(\rho{\downarrow_v})$. Next, for each non-root variable $v \in V$ with $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$, consider the graph $G'_e(v)$ constructed with respect to $\tau_{w_1}, \ldots, \tau_{w_k}$. Let $\{a_1, \ldots, a_{|\rho{\downarrow_v}|}\}$ be the actions of $\rho{\downarrow_v}$ numbered in the order of their appearance along $\rho{\downarrow_v}$. Let $\{y_{w_1}(1), \ldots, y_{w_k}(1)\}$ denote the prevail condition of a_1 with each $y_{w_i}(1)$ being time-stamped with its earliest appearance along τ_{w_i}, that is, $y_{w_i}(1) \in \{\mathsf{b}^1_{w_i}, \mathsf{w}^1_{w_i}\}$. Now, for $2 \leq j \leq |\rho{\downarrow_v}|$, we set $\{y_{w_1}(j), \ldots, y_{w_k}(j)\}$ to the prevail condition of a_i with each $y_{w_i}(j)$ being time-stamped with the lowest possible time index along τ_{w_i} satisfying "$y_{w_i}(j-1)$ does not come after $y_{w_i}(j)$ along τ_{w_i}". Given that

(i) $\rho{\downarrow_v}$ is a complete order-preserving restriction of ρ to the v-changing actions A_v,

(ii) the sequence of time-stamped prevail conditions $\{\{y_{w_1}(j), \ldots, y_{w_k}(j)\}\}_{j=1}^{|\rho{\downarrow_v}|}$ is constructed as above, and

(iii) $|\rho{\downarrow_v}| = |\tau_v| - 1$ by the construction of $\overline{x}_\rho$,

we have that $G'_e(v)$ contains a path

$$\langle \mathsf{b}^1_{w_1} \cdots \mathsf{b}^1_{w_k} \rangle \rightarrow \langle y_{w_1}(1) \cdots y_{w_k}(1) \rangle \rightarrow \ldots \rightarrow \langle y_{w_1}(|\rho{\downarrow_v}|) \cdots y_{w_k}(|\rho{\downarrow_v}|) \rangle$$

and the cost of this path is $cost(\rho{\downarrow_v}) < \infty$. However, from the constructive definition of φ_v in the algorithm polytree-k-indegree, we have $\varphi_v(\overline{x}_\rho)$ being the cost of the minimal-cost path of $|\tau_v| - 1$ edges in $G'_e(v)$ originated in $\langle \mathsf{b}^1_{w_1} \cdots \mathsf{b}^1_{w_k} \rangle$, and thus $\varphi_v(\overline{x}_\rho) \leq cost(\rho{\downarrow_v})$. The latter argument is valid for all planning variables $v \in V$, and thus we have

$$\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}_\rho) \;\; \leq \;\; \sum_{v \in V} cost(\rho{\downarrow_v}) \;\; = \;\; cost(\rho),$$

which is what we had to prove. ∎

procedure polytree-k-indegree($\Pi = \langle V, A, I, G, cost \rangle$)
takes a planning task $\Pi \in \mathbf{P}_b$
returns an optimal plan for Π if solvable, and fails otherwise
create a set of variables $\mathcal{X}$ and set their domains as in Eq. A.1
create a set of functions $\mathcal{F} = \{\varphi_v \mid v \in V\}$ with scopes as in Eq. A.2
for each $v \in V$ **do**
  if $\mathsf{pred}(v) = \emptyset$ **then**
    specify φ_v according to Eq. A.3
  elseif $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$ **then**
    construct graph $G(v)$
    for each k-tuple $\tau'_1 \in \trianglerighteq^*[\tau(w_1)], \ldots, \tau'_k \in \trianglerighteq^*[\tau(w_k)]$ **do**
      construct graph $G'(v)$ from graph $G(v)$ and sequences $\tau'_1, \ldots, \tau'_k$
      construct graph $G'_e(v)$ from graph $G'(v)$
      for each goal-valid sequence $\tau' \in \trianglerighteq^*[\tau(v)]$ **do**
        $\pi :=$ minimal-cost path of $|\tau'| - 1$ edges
          from the source node $\langle \mathsf{b}_{w_1} \cdots \mathsf{b}_{w_k} \rangle$ of $G'_e(v)$
        if returned π **then**
          $\varphi_v(\tau', \tau'_1, \ldots, \tau'_k) := cost(\pi)$
        else
          $\varphi_v(\tau', \tau'_1, \ldots, \tau'_k) := \infty$
        endif
      endfor
    endfor
  endif
endfor
set $\mathsf{COP}_\Pi := (\mathcal{X}, \mathcal{F})$ with global objective min $\sum_{\varphi \in \mathcal{F}} \varphi(\mathcal{X})$
$\overline{x} :=$ solve-tree-cop(COP_Π)
if $\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}) = \infty$ **then return** failure
extract plan ρ from $\overline{x}$ with $cost(\rho) = \sum_{\varphi \in \mathcal{F}} \varphi(\overline{x})$
return ρ

Figure A.3: Algorithm for cost-optimal planning for $\mathbf{P}_b$.

Theorem 23 *Cost-optimal planning for* $\mathbf{P}_b$ *is tractable.*

Proof: The correctness of the polytree-k-indegree algorithm is given by Lemma 9. We now show that, given a planning task $\Pi \in \mathbf{P}_b$, the corresponding constraint optimization problem COP_Π can be constructed and solved in time polynomial in the description size of Π.

Let n be the number of state variables in Π, and κ be the maximal node in-degree in the causal graph $CG(\Pi)$. In polytree-k-indegree, for each planning variable $v \in V$ with $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$, and each k-tuple $\tau'_1 \in \trianglerighteq^*[\tau(w_1)], \ldots, \tau'_k \in \trianglerighteq^*[\tau(w_k)]$, we

(i) construct the graph $G'_e(v)$, and

(ii) use the Dijkstra algorithm to compute shortest paths from the source node of $G'_e(v)$ to all other nodes in that graph.

For each w_i, we have $\tau(w_i) = n$, and thus the number of k-tuples as above for each $v \in V$ is $O(n^k)$. For each such k-tuple, the corresponding graph $G'_e(v)$ can be constructed in time linear in the number of its edges $= O(n^{2k+2} \cdot |A_v|^2) = O(n^{2k+2} \cdot 2^{2k+2})$ (Brafman & Domshlak, 2003). The time complexity of the Dijkstra algorithm on a digraph $G = (N, E)$ is $O(E \log(N))$, and on $G'_e(v)$ it gives us $O\left(n^{2k+2} \cdot 2^{2k+2} \cdot \log\left(n^{k+1} \cdot 2^{k+1}\right)\right)$. Putting things together, the complexity of constructing COP_Π is

$$O\left(n^{3\kappa+3} \cdot 2^{2\kappa+2} \cdot \log\left(n^{\kappa+1} \cdot 2^{\kappa+1}\right)\right). \tag{A.10}$$

Applying a tree-decomposition of COP_Π along the scopes of its functional components we arrive into an equivalent, tree-structured constraint optimization problem over n variables with domains of size $O(n^{\kappa+1})$. This COP is defined by the hard binary "compatibility" constraints between the variables, and costs associated with the variables' values. Such a tree-structured COP can be solved in time $O(xy^2)$ where x is the number of variables and y is an upper bound on the size of a variable's domain (Dechter, 2003). Therefore, solving our COP_Π can be done in time $O(n^{2\kappa+3})$. As the expression in Eq. A.10 dominates both $O(n^{2\kappa+3})$, and the time complexity of extracting a plan from the optimal solution to COP_Π (see the proof of (I) in Lemma 9), the overall complexity of the algorithm polytree-k-indegree is given by Eq. A.10. And since in $\mathbf{P}_b$ we have $\kappa = O(1)$, we conclude that the complexity of polytree-k-indegree is polynomial in the description size of Π.

■

A.2 Cost-Optimal Planning for P(1) with Uniform-Cost Actions

In this section we provide a polynomial time algorithm for cost-optimal planning for $\mathbf{P}(1)$ problems with uniform-cost actions. We begin with showing that such problems exhibit an interesting property, then we exploit this property for devising a planning-to-COP scheme for these problems, and then prove the correctness and complexity of the algorithm.

Here as well, we begin with providing some notation. Given a $\mathbf{P}(1)$ planning task $\Pi = \langle V, A, I, G, cost\rangle$, for each $v \in V$, each $w \in \mathsf{pred}(v)$, and each $\alpha \in \{\mathsf{b}_v, \mathsf{w}_v\}$, $\beta \in \{\mathsf{b}_w, \mathsf{w}_w\}$, by $a_{\alpha|\beta}$ we denote the action a with $\mathsf{eff}(a)[v] = \alpha$ and $\mathsf{pre}(a)[w] = \beta$. Since Π is 1-dependent, the applicability of $a_{\alpha|\beta}$ is prevailed only by the value of w. It is important to keep in mind that $a_{\alpha|\beta}$ is just a notation; the action $a_{\alpha|\beta}$ may *not* belong to the action set A of Π.

A.2.1 Post-Unique Plans and P(1) Problems

We now proceed with introducing the notion of post-unique action sequences that plays a key role in our planning-to-COP compilation here.

Definition 21 *Let $\Pi = \langle V, A, I, G, cost\rangle$ be a* UB *planning task. An action sequence $\varrho \in A^*$ is called* **post-unique** *if, for each pair of actions $a, a' \in \varrho$, we have $\mathsf{eff}(a) = \mathsf{eff}(a')$ only if $a = a'$. That is, all the changes of each variable to a certain value along ϱ are performed by the same (type of) action. The (possibly empty) set of all* **post-unique plans** *for Π is denoted by $\mathcal{P}^c(\Pi)$ (or simply $\mathcal{P}^c$, if the identity of Π is clear from the context).*

The notion of post-unique action sequences is closely related to the notion of post-unique planning problems (Bäckström & Klein, 1991; Bäckström & Nebel, 1995), but is considerably weaker than the latter. While action sets of post-unique planning problems are not allowed to contain two actions with the same effect, Definition 21 poses a similar restriction only on action sequences, and

not on the underlying planning problems. Still, the property of post-uniqueness for plans is strong. In general, solvable problems in UB may not exhibit post-unique plans at all. Turns out, however, that for the problems in $\mathbf{P}(1)$ this is very much not the case.

Theorem 24 *For any solvable* $\mathbf{P}(1)$ *task* $\Pi = \langle V, A, I, G, cost \rangle$*, we have* $\mathcal{P}^c(\Pi) \neq \emptyset$*. Moreover, if the actions* A *are uniform-cost, then* $\mathcal{P}^c(\Pi)$ *contains at least one cost-optimal plan.*

Proof: As the correctness of the second claim immediately implies the correctness of the first one, here we focus on the proof the second claim. Given a $\mathbf{P}(1)$ planning task $\Pi = \langle V, A, I, G, cost \rangle$ with uniform-cost actions, and plan $\rho = \langle a_1, \ldots, a_m \rangle$ for Π, we construct a sequence of actions ρ^*

- ρ^* is a post-unique plan for Π,
- $cost(\rho^*) = cost(\rho)$.

This construction is two-step. First, for each $v \in V$, we map the subsequence $\rho{\downarrow_v} = \langle a_{i_1}, \ldots, a_{i_k} \rangle$ into a post-unique sequence of actions $\rho^*_v = \langle a^*_{i_1}, \ldots, a^*_{i_k} \rangle$. Note that the indexes $i_1, \ldots, i_k$ of the action elements of each $\rho{\downarrow_v}$ are the *global* indexes of these actions along ρ, and *exactly the same* indexes are used for marking the elements of the constructed sequences ρ^*_v. Having constructed the sequences $\rho^*_{v_1}, \ldots, \rho^*_{v_n}$, we then merge them into a single actions sequence ρ^*, and show that ρ^* is a valid plan for Π. The two properties of ρ^* as required above will then hold immediately because $|\rho^*| = |\rho|$, and post-uniqueness of ρ^* is implied by the individual post-uniqueness of all its per-variable components ρ^*_v.

The mapping of subsequences $\rho{\downarrow_v}$ of ρ to the desired sequences ρ^*_v for all variables v is performed top-down, consistently with a topological ordering of the causal graph $CG(\Pi)$. This top-down processing allows us to assume that, when constructing ρ^*_v, the subsequences ρ^*_w for all $w \in \mathsf{pred}(v)$ are already constructed. Given that, while mapping each $\rho{\downarrow_v} = \langle a_{i_1}, \ldots, a_{i_k} \rangle$ to the corresponding ρ^*_v, we distinguish between the following three cases.

(1) *The subsequence* $\rho{\downarrow_v}$ *is already post-unique.*

In this case, we simply set ρ^*_v to $\rho{\downarrow_v}$. In addition, we construct the following sets of ordering constraints. First, we set a binary relation $>_v$ over the action elements of $\rho^*_v = \langle a^*_{i_1}, \ldots, a^*_{i_k} \rangle$ to

$$>_v = \{a^*_i < a^*_j \mid a^*_i, a^*_j \in \rho^*_v, i < j\}. \tag{A.11}$$

It is immediate from Eq. A.11 that $>_v$ is a strict total order over the elements of ρ^*_v as $>_v$ simply follows the action indexing inherited by ρ^*_v from plan ρ via $\rho{\downarrow_v}$.

Now, for each $w \in \mathsf{pred}(v)$, we set a binary relation $>_{v,w}$ over the elements of ρ^*_v and ρ^*_w to

$$>_{v,w} = \begin{cases} \bigcup_{a^*_i \in \rho^*_v, a^*_j \in \rho^*_w} \{a^*_i < a^*_j \mid i < j\} \cup \{a^*_j < a^*_i \mid j < i\}, & \mathsf{pre}(a)[w] \text{ is specified for some } a \in \rho^*_v \\ \emptyset, & \text{otherwise} \end{cases}. \tag{A.12}$$

For each $w \in \mathsf{pred}(v)$, the relation $>_{v,w}$ defined by Eq. A.12 is a strict total order over its domain because the ordering constraints between the elements of ρ^*_v and ρ^*_w are a *subset* of the constraints induced by the total-order plan ρ over the (corresponding) actions from $\rho{\downarrow_v}$ and $\rho{\downarrow_w}$. For the same reason, from Eqs. A.11 and A.12, we have that, for each $w \in \mathsf{pred}(v)$, $>_v \cup >_{v,w}$ is a strict total order over the union of the elements of ρ^*_v and ρ^*_w.

From Eqs. A.11-A.12 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho^*_v| = |\rho{\downarrow_v}|$ implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(2) *The subsequence $\rho{\downarrow_v}$ is not post-unique, but the actions in $\rho{\downarrow_v}$ are all prevailed by the value of a single parent $w \in \mathsf{pred}(v)$.*

Since $\rho{\downarrow_v}$ is *not* post-unique, $\rho{\downarrow_v}$ in this case has to contain instances of at least three action types from $\{a_{\mathsf{b}_v|\mathsf{b}_w}, a_{\mathsf{b}_v|\mathsf{w}_w}, a_{\mathsf{w}_v|\mathsf{b}_w}, a_{\mathsf{w}_v|\mathsf{w}_w}\}$. Thus, in particular, it must be that

(a) $|\rho w{\downarrow_|} \geq 1$, and

(b) for some $\beta \in \{\mathsf{b}_w, \mathsf{w}_w\}$, we have $a_{\mathsf{w}_v|\beta}, a_{\mathsf{b}_v|\beta} \in \rho{\downarrow_v}$.

Given that, we set $\rho^*_v = \langle a^*_{i_1}, \ldots, a^*_{i_k}\rangle$ to

$$\forall 1 \leq j \leq k: \;\; a^*_{i_j} = \begin{cases} a_{\mathsf{w}_v|\beta}, & j \text{ is odd} \\ a_{\mathsf{b}_v|\beta}, & j \text{ is even} \end{cases}.$$

Both post-uniqueness of such ρ^*_v, as well as its applicability with respect to v are straightforward. The ordering constraints $>_v$ are then set according to Eq. A.11. Likewise, if $\rho^*_w = \langle a_{j_1}, \ldots, a_{j_l}\rangle$, we set

$$>_{v,w} = \begin{cases} \bigcup_{a^*_i \in \rho^*_v} \{a^*_i < a_{j_1}\}, & \beta = \mathsf{b}_w \\ \bigcup_{a^*_i \in \rho^*_v} \{a^*_i > a_{j_1}\}, & \beta = \mathsf{w}_w, l = 1 \\ \bigcup_{a^*_i \in \rho^*_v} \{a^*_i > a_{j_1}\} \cup \{a^*_i < a_{j_2}\}, & \beta = \mathsf{w}_w, l > 1 \end{cases} \tag{A.13}$$

Finally, the ordering constraints $>_{v,w'}$ for the rest of the parents $w' \in \mathsf{pred}(v) \setminus \{w\}$ are set to empty sets.

The relation $>_v$ here is identical to this in case (1), and thus it is a strict total order over the elements of ρ^*_v. From Eq. A.13, it is easy to verify that $>_{v,w}$ is also a strict partial order over the union of the elements of ρ^*_v and ρ^*_w. Finally, as all the elements of ρ^*_v are all identically constrained with respect to the elements of ρ^*_w, we have $>_v \cup >_{v,w}$ forming a strict partial order over the union of the elements of ρ^*_v and ρ^*_w. (For all other parents $w' \in \mathsf{pred}(v)$, we simply have $>_v \cup >_{v,w} = >_v$.)

From Eqs. A.11 and A.13 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho^*_v| = |\rho{\downarrow_v}|$ implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(3) *The subsequence $\rho{\downarrow_v}$ is not post-unique, and the actions of $\rho{\downarrow_v}$ are prevailed by more than one parent of v.*

The setting of this case in particular implies that there is a pair of v's parents $\{u, w\} \subseteq \mathsf{pred}(v)$ such that $a_{\mathsf{w}_v|\alpha}, a_{\mathsf{b}_v|\beta} \in \rho_v$ for some $\alpha \in \{\mathsf{b}_u, \mathsf{w}_u\}$, $\beta \in \{\mathsf{b}_w, \mathsf{w}_w\}$. Given that, we set ρ^*_v to

$$\forall 1 \leq j \leq k: \;\; a^*_{i_j} = \begin{cases} a_{\mathsf{w}_v|\alpha}, & j \text{ is odd} \\ a_{\mathsf{b}_v|\beta}, & j \text{ is even} \end{cases},$$

and, similarly to case (2), both post-uniqueness of ρ^*_v, and its applicability with respect to v are straightforward.

Here as well, the ordering constraints $>_v$ are set according to Eq. A.11. Likewise, if $\rho^*_w = \langle a_{j_1}, \ldots, a_{j_l} \rangle$, and $\rho^*_u = \langle a_{j'_1}, \ldots, a_{j'_{l'}} \rangle$, we set $>_{v,w}$ according to Eq. A.13 above, and $>_{v,u}$ according to Eq. A.14 below.

$$>_{v,u} = \begin{cases} \bigcup_{a^*_i \in \rho^*_v} \{a^*_i < a_{j'_1}\}, & \alpha = \mathsf{b}_u \\ \bigcup_{a^*_i \in \rho^*_v} \{a^*_i > a_{j'_1}\}, & \alpha = \mathsf{w}_u, l' = 1 \\ \bigcup_{a^*_i \in \rho^*_v} \{a^*_i > a_{j'_1}\} \cup \{a^*_i < a_{j'_2}\}, & \alpha = \mathsf{w}_u, l' > 1 \end{cases} \tag{A.14}$$

Finally, the ordering constraints $>_{v,w'}$ for the rest of the parents $w' \in \mathsf{pred}(v) \setminus \{u, w\}$ are set to empty sets.

The relation $>_v$ here is identical to this in cases (1-2), and relations $>_{v,u}$ and $>_{v,w}$ are effectively identical to the relation $>_{v,w}$ in case (2). Thus, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ^*_v and ρ^*_u, and ρ^*_v and ρ^*_w, respectively.

From Eqs. A.11, A.13, and A.14 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho^*_v| = |\rho{\downarrow_v}|$ implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

As the last step, we now prove that, for each $v \in V$ and each $w \in \mathsf{pred}(v)$, we have $>_w \cup >_{v,w}$ being a strict partial order over the union of the elements of ρ^*_w and ρ^*_v.

- If $>_{v,w}$ is constructed according to Eq. A.12, then $>_w \cup >_{v,w}$ is a subset of the constraints induced by plan ρ over the (corresponding to ρ^*_v and ρ^*_w) actions from ρ_v and ρ_w.
- Otherwise, if $>_{v,w}$ is constructed according to Eq. A.13 or (for us here identical) Eq. A.14, then $>_{v,w}$ (i) addresses at most two elements of ρ_w, (ii) orders these elements consistently with $>_w$.

In both cases, the argued properties of $>_w \cup >_{v,w}$ implies that it forms a strict partial order over the union of the elements of ρ^*_v and ρ^*_w.

Until now, we have specified the sequences ρ^*_v, the orders $>_v$ induced by these sequences, the orders $>_{v,w}$, and proved that all $>_v \cup >_{v,w}$ and $>_w \cup >_{v,w}$ form strict partial orders over their domains. This construction allows us to apply now Theorem 22 to the (considered as sets) sequences ρ^*_v and orders $>_v$ and $>_{v,w}$, proving that

$$> = \bigcup_{v \in V} (>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w})$$

forms a strict partial order over the union of $\rho^*_{v_1}, \ldots, \rho^*_{v_n}$. Putting thing together, the above implies that any linearization ρ^* of $>$ is a plan for Π, and post-uniqueness of all its subsequences $\rho^*_{v_1}, \ldots, \rho^*_{v_n}$ then implies $\rho^* \in \mathcal{P}^{\mathrm{c}}(\Pi)$. Moreover, if ρ is an optimal plan for Π, then $|\rho^*| = |\rho|$ implies the optimality of ρ^*. ∎

A.2.2 Construction

The main impact of Theorem 24 on our planning-to-COP scheme for uniform-cost $\mathbf{P}(1)$ is that we can now restrict our attention to post-unique plans only. Given that, the constraint optimization

problem $\mathsf{COP}_\Pi = (\mathcal{X}, \mathcal{F})$ for a uniform-cost planning task $\Pi = \langle V, A, I, G, cost \rangle \in \mathbf{P}(1)$ is specified as follows.

The variable set $\mathcal{X}$ contains a variable x_v for each planning variable $v \in V$, and a variable x_v^w for each edge $(w, v) \in CG(\Pi)$. That is,

$$\begin{aligned} \mathcal{X} &= \mathcal{X}^V \cup \mathcal{X}^{\mathcal{E}}, \\ \mathcal{X}^V &= \{x_v \mid v \in V\} \\ \mathcal{X}^{\mathcal{E}} &= \{x_v^w \mid (w, v) \in CG(\Pi)\} \end{aligned} \tag{A.15}$$

For each variable $x_v \in \mathcal{X}^V$, the domain $\mathcal{D}(x_v)$ consists of all goal-valid prefixes of $\sigma(v)$. For each variable $x_v^w \in \mathcal{X}^{\mathcal{E}}$, the domain $\mathcal{D}(x_v^w)$ consists of all triples of integers $[\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!]$ satisfying Eq. A.16.

$$\begin{aligned} \mathcal{D}(x_v) &= \trianglerighteq^*[\sigma(v)] \\ \mathcal{D}(x_v^w) &= \{[\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!] \mid \delta_{\mathsf{w}}, \delta_{\mathsf{b}} \in \{0, 1\}, 0 \leq \eta \leq n\} \end{aligned} \tag{A.16}$$

The semantics of Eq. A.16 is as follows. Let $\{w_1, \ldots, w_k\}$ be an *arbitrary* fixed ordering of $\mathsf{pred}(v)$. If x_v takes the value $\sigma_v \in \mathcal{D}(x_v)$, then v is forced to provide σ_v sequence of values. In turn, if $x_v^{w_i}$ takes the value $[\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!]$, then η corresponds to the number of value changes of v, $\delta_{\mathsf{w}} = 1$ ($\delta_{\mathsf{b}} = 1$) forces the *subset* of parents $\{w_1, \ldots, w_i\} \subseteq \mathsf{pred}(v)$ to support (that is, prevail) all the changes of v to w_v (respectively, to b_v), and $\delta_{\mathsf{w}} = 0$ ($\delta_{\mathsf{b}} = 0$) relieves this subset of parents $\{w_1, \ldots, w_i\}$ from this responsibility.

For each variable $x \in \mathcal{X}$, the set $\mathcal{F}$ contains a non-negative, real-valued function φ_x with the scope

$$Q_x = \begin{cases} \{x_v\}, & x = x_v, k = 0 \\ \{x_v, x_v^{w_k}\}, & x = x_v, k > 0 \\ \{x_v^{w_1}, x_{w_1}\}, & x = x_v^{w_1}, k > 0 \\ \{x_v^{w_j}, x_v^{w_{j-1}}, x_{w_j}\}, & x = x_v^{w_j}, 1 < j \leq k \end{cases} \tag{A.17}$$

where $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$ (and $k = 0$ means $\mathsf{pred}(v) = \emptyset$). Proceeding now with specifying these functional components $\mathcal{F}$ of COP_Π, first, for each x_v with $\mathsf{pred}(v) = \emptyset$, and for each $\sigma_v \in \trianglerighteq^*[\sigma(v)]$, we set $\varphi_{x_v}(\sigma_v)$ to

$$\varphi_{x_v}(\sigma_v) = \begin{cases} 0, & |\sigma_v| = 1, \\ 1, & (|\sigma_v| = 2) \wedge (a_{\mathsf{w}_v} \in A_v), \\ |\sigma_v| - 1, & (|\sigma_v| > 2) \wedge (a_{\mathsf{w}_v}, a_{\mathsf{b}_v} \in A_v), \\ \infty, & \text{otherwise} \end{cases} \tag{A.18}$$

In turn, for each planning variable $v \in V$ with $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$, $k > 0$, the function φ_{x_v} is set to

$$\varphi_{x_v}(\sigma_v, [\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!]) = \begin{cases} 0, & (|\sigma_v| = 1) \wedge ([\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!] = [\![0, 0, 0]\!]), \\ 1, & (|\sigma_v| = 2) \wedge ([\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!] = [\![1, 0, 1]\!]), \\ |\sigma_v| - 1, & (|\sigma_v| > 2) \wedge ([\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!] = [\![1, 1, |\sigma_v| - 1]\!]), \\ \infty, & \text{otherwise} \end{cases} \tag{A.19}$$

The functions φ_{x_v} capture the, *marginal over the actions* A_v, cost of providing a sequence σ_v of value changes of v in Π, given that (in case of Eq. A.19) the parents of v are "ready to support these value changes". In specifying the remaining functional components we use an "indicator" function φ specified in Eq. A.20.

$$\varphi\left([\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!], \sigma_w\right) = \begin{cases} 0, & \delta_{\mathsf{w}} = 0, \delta_{\mathsf{b}} = 0, \\ 0, & \delta_{\mathsf{w}} = 1, \delta_{\mathsf{b}} = 0, (a_{\mathsf{w}_v|\mathsf{b}_w} \in A_v) \vee ((|\sigma_w| > 1) \wedge (a_{\mathsf{w}_v|\mathsf{w}_w} \in A_v)), \\ 0, & \delta_{\mathsf{w}} = 0, \delta_{\mathsf{b}} = 1, (a_{\mathsf{b}_v|\mathsf{b}_w} \in A_v) \vee ((|\sigma_w| > 1) \wedge (a_{\mathsf{b}_v|\mathsf{w}_w} \in A_v)), \\ 0, & \delta_{\mathsf{w}} = 1, \delta_{\mathsf{b}} = 1, (a_{\mathsf{w}_v|\mathsf{b}_w}, a_{\mathsf{b}_v|\mathsf{b}_w} \in A_v) \vee ((|\sigma_w| > 1) \wedge (a_{\mathsf{w}_v|\mathsf{w}_w}, a_{\mathsf{b}_v|\mathsf{w}_w} \in A_v)), \\ 0, & \delta_{\mathsf{w}} = 1, \delta_{\mathsf{b}} = 1, |\sigma_w| \geq \eta, a_{\mathsf{w}_v|\mathsf{b}_w}, a_{\mathsf{b}_v|\mathsf{w}_w} \in A_v, \\ 0, & \delta_{\mathsf{w}} = 1, \delta_{\mathsf{b}} = 1, |\sigma_w| > \eta, a_{\mathsf{w}_v|\mathsf{w}_w}, a_{\mathsf{b}_v|\mathsf{b}_w} \in A_v, \\ \infty, & \text{otherwise} \end{cases} \tag{A.20}$$

The semantics of φ is that, for each planning variable $v \in V$, each $w \in \mathsf{pred}(v)$, and each $([\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!], \sigma_w) \in \mathcal{D}(x_v^w) \times \mathcal{D}(x_w)$, we have $\varphi([\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!], \sigma_w) = 0$ if the value sequence σ_w of w can support *all* the changes of v to w_v (if $\delta_{\mathsf{w}} = 1$) and *all* the changes of v to b_v (if $\delta_{\mathsf{b}} = 1$), out of η value changes of v in Π. Given this indicator function φ, for each $v \in V$, the functional component $\varphi_{x_v^{w_1}}$ is specified as

$$\varphi_{x_v^{w_1}}\left([\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!], \sigma_{w_1}\right) = \varphi\left([\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!], \sigma_{w_1}\right), \tag{A.21}$$

and the rest of the functions $\varphi_{x_v^{w_2}}, \ldots, \varphi_{x_v^{w_k}}$ are specified as follows. For each $2 \leq i \leq k$, the value of the function $\varphi_{x_v^{w_j}}$ at the combination of $[\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!] \in \mathcal{D}(x_v^{w_i})$, $[\![\delta'_{\mathsf{w}}, \delta'_{\mathsf{b}}, \eta']\!] \in \mathcal{D}(x_v^{w_{i-1}})$, and $\sigma_{w_i} \in \mathcal{D}(x_{w_i}) = \trianglerighteq^*[\sigma(w_i)]$ is specified as

$$\varphi_{x_v^{w_i}}\left([\![\delta_{\mathsf{w}}, \delta_{\mathsf{b}}, \eta]\!], [\![\delta'_{\mathsf{w}}, \delta'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}\right) = \begin{cases} \varphi\left([\![\delta_{\mathsf{w}} - \delta'_{\mathsf{w}}, \delta_{\mathsf{b}} - \delta'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}\right), & \eta = \eta' \wedge \delta_{\mathsf{w}} \geq \delta'_{\mathsf{w}} \wedge \delta_{\mathsf{b}} \geq \delta'_{\mathsf{b}} \\ \infty & \text{otherwise} \end{cases} \tag{A.22}$$

This finalized the construction of COP_Π, and this construction constitutes the first three steps of the algorithm polytree-1-dep-uniform in Figure A.4(a). The subsequent steps of this algorithm are conceptually similar to these of the polytree-k-indegree algorithm in Section A.1, with the major difference being in the plan reconstruction routines. It is not hard to verify from Eqs. A.15-A.17, and the fact that the causal graph of $\Pi \in \mathbf{P}(1)$ forms a polytree that

(i) for each variable $x \in \mathcal{X}$, $|\mathcal{D}(x)| = poly(n)$,

(ii) the tree-width of the cost network of $\mathcal{F}$ is ≤ 3, and

(iii) the optimal tree-decomposition of the COP_Π's cost network is given by any topological ordering of the causal graph that is consistent with the (arbitrary yet fixed at the time of the COP_Π's construction) orderings of each planning variable's parents in the causal graph.

For an illustration, we refer the reader to Figure 3.3 (pp. 23) where Figure 3.3(a) depicts the causal graph of a planning task $\Pi \in \mathbf{P}(1)$, and Figure 3.3(c) depicts the cost network of the corresponding COP_Π. The top-most variables and the cliques in the cost network correspond to the functional components of COP_Π.

procedure polytree-1-dep-uniform($\Pi = \langle V, A, I, G, cost \rangle$)
takes a planning task $\Pi \in \mathbf{P}(1)$ with uniform-cost actions A
returns a cost-optimal plan for Π if Π is solvable, and fails otherwise
create a set of variables $\mathcal{X}$ as in Eqs. A.15-A.16
create a set of functions $\mathcal{F} = \{\varphi_x \mid x \in \mathcal{X}\}$ with scopes as in Eq. A.17
for each $x \in \mathcal{X}$ **do**
    specify φ_x according to Eqs. A.18-A.22
endfor
set $\mathsf{COP}_\Pi := (\mathcal{X}, \mathcal{F})$ with global objective min $\sum_{\varphi \in \mathcal{F}} \varphi(\mathcal{X})$
$\overline{x} :=$ solve-tree-cop(COP_Π)
if $\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}) = \infty$ **then return** failure
extract plan ρ from $\overline{x}$ with $cost(\rho) = \sum_{\varphi \in \mathcal{F}} \varphi(\overline{x})$
return ρ

Figure A.4: Algorithm for cost-optimal planning for $\mathbf{P}(1)$ tasks with uniform-cost actions.

A.2.3 Correctness and Complexity

Lemma 10 *Let Π be a $\mathbf{P}(1)$ task with uniform-costs actions, $\mathsf{COP}_\Pi = (\mathcal{X}, \mathcal{F})$ be the corresponding constraint optimization problem, and $\overline{x}$ be an optimal assignment to $\mathcal{X}$ with $\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}) = \alpha$.*

(I) If $\alpha < \infty$, then a plan of cost α for Π can be reconstructed from $\overline{x}$ in time polynomial in the description size of Π.

(II) If Π has a plan, then $\alpha < \infty$.

Proof:
(I) Given a COP solution $\overline{x}$ with $\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}) = \alpha < \infty$, we construct a plan ρ for Π with $cost(\rho) = \alpha$. We construct this plan by

1. Traversing the planning variables in a topological ordering of the causal graph $CG(\Pi)$, and associating each variable v with a sequence $\rho_v \in A_v^*$.

2. Merging the constructed sequences $\rho_{v_1}, \ldots, \rho_{v_n}$ into the desired plan ρ.

For each variable $x_v \in \mathcal{X}$, let σ_v denote the value provided by $\overline{x}$ to x_v. First, for each variable $v \in V$ with $\mathsf{pred}(v) = \emptyset$, let a sequence ρ_v of actions from A_v be defined as

$$\rho_v = \begin{cases} \emptyset & |\sigma_v| = 1 \\ a_v^1 \cdot \ldots \cdot a_v^{|\sigma_v|-1} & \text{otherwise} \end{cases}, \tag{A.23}$$

where, for $1 \leq j \leq |\sigma_v| - 1$,

$$a_v^j = \begin{cases} a_{\mathsf{b}_v}, & j \text{ is even} \\ a_{\mathsf{w}_v}, & j \text{ is odd} \end{cases}, \tag{A.24}$$

with $\mathsf{eff}(a_{\mathsf{b}_v}) = \{\mathsf{b}_v\}$, and $\mathsf{eff}(a_{\mathsf{w}_v}) = \{\mathsf{w}_v\}$. From Eq. A.18 and $\varphi_v(\sigma_v) \leq \alpha < \infty$, we immediately have (i) $\{a_{\mathsf{w}_v}\} \subseteq A_v$ if $|\sigma_v| \geq 2$, and $\{a_{\mathsf{b}_v}, a_{\mathsf{w}_v}\} \subseteq A_v$ if $|\sigma_v| > 2$, and (ii) $cost(\rho_v) = \varphi_v(\sigma_v)$.

Let a binary relation $>_v$ over the action elements of ρ_v be defined as the transitive closure of $\{a_v^{j-1} < a_v^j \mid 1 < j \leq |\sigma_v| - 1\}$, that is

$$>_v = \{a_v^{j'} < a_v^j \mid 1 \leq j' < j \leq |\sigma_v| - 1\} \tag{A.25}$$

Clearly, $>_v$ constitutes a strict total ordering over the elements of ρ_v, making ρ_v an applicable sequence of actions that provides to v the value $G[v]$ if the latter is specified.

Next, for each variable $v \in V$ with $\mathsf{pred}(v) \neq \emptyset$, let $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$ be numbered according to their ordering used for constructing COP_Π. Likewise, for each $w_i \in \mathsf{pred}(v)$, let $[\![\delta_{\mathsf{w}}(i), \delta_{\mathsf{b}}(i), \eta(i)]\!]$ be the value provided by $\overline{x}$ to $x_v^{w_i}$. Given that, let a pair of indexes $0 \leq \langle \mathsf{w} \rangle, \langle \mathsf{b} \rangle \leq k$ be defined as

$$\langle \mathsf{w} \rangle = \begin{cases} 0, & \delta_{\mathsf{w}}(k) = 0, \\ 1, & \delta_{\mathsf{w}}(1) = 1, \\ j, & \delta_{\mathsf{w}}(j-1) < \delta_{\mathsf{w}}(j),\ 2 \leq j \leq k \end{cases} \tag{A.26}$$

$$\langle \mathsf{b} \rangle = \begin{cases} 0, & \delta_{\mathsf{b}}(k) = 0, \\ 1, & \delta_{\mathsf{b}}(1) = 1, \\ j, & \delta_{\mathsf{b}}(j-1) < \delta_{\mathsf{b}}(j),\ 2 \leq j \leq k \end{cases} \tag{A.27}$$

Informally, in our next-coming construction of the action sequence ρ_v for the state variable v, $\langle \mathsf{w} \rangle$ and $\langle \mathsf{b} \rangle$ will indicate the parents prevailing the value changes of v to w_v and to b_v, respectively, along ρ_v. Note that Eqs. A.26-A.27 are well-defined because, for $2 \leq j \leq k$, Eq. A.22 implies

$$\delta_{\mathsf{w}}(j-1) \leq \delta_{\mathsf{w}}(j) \ \ \wedge \ \ \delta_{\mathsf{b}}(j-1) \leq \delta_{\mathsf{b}}(j) \ \ \wedge \ \ \eta(j-1) = \eta(j).$$

Given this notation, the action sequence ρ_v and the partial orders $>_{v,w_1}, \ldots, >_{v,w_k}$ are constructed as follows.

[$\langle \mathsf{w} \rangle = 0, \langle \mathsf{b} \rangle = 0$] In this case, the constructed plan ρ should perform no value changes of v, and thus ρ_v is set to an empty action sequence, and, consequently, both $>_v$ and all $>_{v,w}$ are set to empty sets.

[$\langle \mathsf{w} \rangle > 0, \langle \mathsf{b} \rangle = 0$] In this case, the constructed plan ρ should perform exactly one value change of v (from b_v to w_v), and thus ρ_v is set to contain exactly one action a_v^1 with $\mathsf{eff}(a) = \{\mathsf{w}_v\}$, and

$$\mathsf{pre}(a_v^1) = \begin{cases} \{\mathsf{b}_v, \mathsf{b}_{w_{\langle \mathsf{w} \rangle}}\}, & a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}} \in A_v \\ \{\mathsf{b}_v, \mathsf{w}_{w_{\langle \mathsf{w} \rangle}}\}, & \text{otherwise} \end{cases} \tag{A.28}$$

Note that a_v^1 is well-defined, as $\alpha < \infty$ and Eq. A.20 together imply that $\{a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}, a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}\} \cap A_v \neq \emptyset$ (see case (2) in Eq. A.20). In both outcomes of Eq. A.28, we set $>_v = \emptyset$. If $a_v^1 = a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}$, we set

$$>_{v,w_{\langle \mathsf{w} \rangle}} = \{a_v^1 < a_{w_{\langle \mathsf{w} \rangle}}^1 \mid a_{w_{\langle \mathsf{w} \rangle}}^1 \in \rho_{w_{\langle \mathsf{w} \rangle}}\} \tag{A.29}$$

Otherwise, if $a_v^1 = a_{\mathsf{w}_v|\mathsf{w}_{w_{\langle \mathsf{w} \rangle}}}$, then from case (2) in Eq. A.20, $a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}} \notin A_v$, and $\alpha < \infty$, we have $|\sigma_{w_{\langle \mathsf{w} \rangle}}| > 1$, and thus $|\rho_{w_{\langle \mathsf{w} \rangle}}| \geq 1$. Given that, we set

$$>_{v,w_{\langle \mathsf{w} \rangle}} = \{a_{w_{\langle \mathsf{w} \rangle}}^1 < a_v^1\} \cup \{a_v^1 < a_{w_{\langle \mathsf{w} \rangle}}^2 \mid a_{w_{\langle \mathsf{w} \rangle}}^2 \in \rho_{w_{\langle \mathsf{w} \rangle}}\} \tag{A.30}$$

In both cases, it is easy to verify that $>_v \cup >_{v,w_{\langle \mathsf{w} \rangle}} \cup >_{w_{\langle \mathsf{w} \rangle}}$ constitutes a strict total order over the action elements of ρ_v and $\rho_{w_{\langle \mathsf{w} \rangle}}$. (In particular, this trivially implies that $>_v \cup >_{v,w}$ and $>_{v,w} \cup >_w$ are strict partial orderings over their domains.)

From Eqs. A.25, A.29, and A.30 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, Eq. A.16 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

[$\langle \mathsf{w} \rangle > 0, \langle \mathsf{b} \rangle > 0, \langle \mathsf{w} \rangle = \langle \mathsf{b} \rangle$] In this case, the constructed plan ρ should perform more than one value change of v, and all these value changes should be performed by (a pair of types of) actions prevailed by the value of $w_{\langle \mathsf{w} \rangle}$. From $\alpha < \infty$, we have $\varphi([\![\delta_{\mathsf{w}}(\langle \mathsf{w} \rangle), \delta_{\mathsf{b}}(\langle \mathsf{w} \rangle), \eta]\!], \sigma_{w_{\langle \mathsf{w} \rangle}}) = 0$. The specification of the case in question (that is, $\langle \mathsf{w} \rangle = \langle \mathsf{b} \rangle > 0$) thus implies that one of the conditions of the cases (4-6) of Eq. A.20 should hold. Given that, we distinguish between the following four settings.

(1) If $\{a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}, a_{\mathsf{b}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}\} \subseteq A_v$, then ρ_v is specified according to Eq. A.23, and its action elements are specified as

$$a_v^i = \begin{cases} a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}, & i \text{ is odd} \\ a_{\mathsf{b}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}, & i \text{ is even} \end{cases}. \tag{A.31}$$

The relation $>_v$ is set according to Eq. A.25, and $>_{v,w_{\langle \mathsf{w} \rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{w} \rangle}} = \{a_v^i < a_{w_{\langle \mathsf{w} \rangle}}^1 \mid a_v^i \in \rho_v, a_{w_{\langle \mathsf{w} \rangle}}^1 \in \rho_{w_{\langle \mathsf{w} \rangle}}\} \tag{A.32}$$

Finally, for all $w \in \mathsf{pred}(v) \setminus \{w_{\langle \mathsf{w} \rangle}\}$, we set $>_{v,w} = \emptyset$.

(2) Otherwise, if $\{a_{\mathsf{w}_v|\mathsf{w}_{w_{\langle \mathsf{w} \rangle}}}, a_{\mathsf{b}_v|\mathsf{w}_{w_{\langle \mathsf{w} \rangle}}}\} \subseteq A_v$ and $|\sigma_{w_{\langle \mathsf{w} \rangle}}| > 1$, then we have $|\rho_{w_{\langle \mathsf{w} \rangle}}| \geq 1$. Given that, we again set ρ_v according to Eq. A.23, but now with its action elements being set as

$$a_v^i = \begin{cases} a_{\mathsf{w}_v|\mathsf{w}_{w_{\langle \mathsf{w} \rangle}}}, & i \text{ is odd} \\ a_{\mathsf{b}_v|\mathsf{w}_{w_{\langle \mathsf{w} \rangle}}}, & i \text{ is even} \end{cases}. \tag{A.33}$$

The relation $>_v$ is set according to Eq. A.25, and $>_{v,w_{\langle \mathsf{w} \rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{w} \rangle}} = \{a_{w_{\langle \mathsf{w} \rangle}}^1 < a_v^i \mid a_v^i \in \rho_v\} \cup \{a_v^i < a_{w_{\langle \mathsf{w} \rangle}}^2 \mid a_v^i \in \rho_v, a_{w_{\langle \mathsf{w} \rangle}}^2 \in \rho_{w_{\langle \mathsf{w} \rangle}}\} \tag{A.34}$$

Finally, here as well, for all $w \in \mathsf{pred}(v) \setminus \{w_{\langle \mathsf{w} \rangle}\}$, we set $>_{v,w} = \emptyset$.

(3) Otherwise, if $\{a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}, a_{\mathsf{b}_v|\mathsf{w}_{w_{\langle \mathsf{w} \rangle}}}\} \subseteq A_v$, and $|\sigma_{w_{\langle \mathsf{w} \rangle}}| \geq |\sigma_v| - 1$, then ρ_v is specified according to Eq. A.23, and its action elements are specified as

$$a_v^i = \begin{cases} a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}, & i \text{ is odd} \\ a_{\mathsf{b}_v|\mathsf{w}_{w_{\langle \mathsf{w} \rangle}}}, & i \text{ is even} \end{cases}. \tag{A.35}$$

The relation $>_v$ is set according to Eq. A.25, and $>_{v,w_{\langle \mathsf{w} \rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{w} \rangle}} = \bigcup_{a_v^i \in \rho_v, a_{w_{\langle \mathsf{w} \rangle}}^j \in \rho_{w_{\langle \mathsf{w} \rangle}}} \{a_v^i < a_{w_{\langle \mathsf{w} \rangle}}^j \mid i \leq j\} \cup \{a_{w_{\langle \mathsf{w} \rangle}}^j < a_v^i \mid i > j\} \tag{A.36}$$

For all $w \in \mathsf{pred}(v) \setminus \{w_{\langle \mathsf{w} \rangle}\}$, we set $>_{v,w} = \emptyset$.

(4) Otherwise, if $\{a_{\mathsf{w}_v|\mathsf{w}_{w_1}}, a_{\mathsf{b}_v|\mathsf{b}_{w_1}}\} \subseteq A_v$, and $|\sigma_{w_{\langle \mathsf{w} \rangle}}| \geq |\sigma_v|$, then ρ_v is specified according to Eq. A.23, and its action elements are specified as

$$a_v^i = \begin{cases} a_{\mathsf{w}_v|\mathsf{w}_{w_{\langle \mathsf{w} \rangle}}}, & i \text{ is odd} \\ a_{\mathsf{b}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}, & i \text{ is even} \end{cases}. \tag{A.37}$$

The relation $>_v$ is set according to Eq. A.25, and $>_{v,w_{\langle \mathsf{w} \rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{w} \rangle}} = \bigcup_{a_v^i \in \rho_v, a_{w_{\langle \mathsf{w} \rangle}}^j \in \rho_{w_{\langle \mathsf{w} \rangle}}} \{a_v^i < a_{w_{\langle \mathsf{w} \rangle}}^j \mid i < j\} \cup \{a_{w_{\langle \mathsf{w} \rangle}}^j < a_v^i \mid i \geq j\} \tag{A.38}$$

For all $w \in \mathsf{pred}(v) \setminus \{w_{\langle \mathsf{w} \rangle}\}$, we set $>_{v,w} = \emptyset$.

In all the four cases above, $>_v \cup >_{v,w_{\langle \mathsf{w} \rangle}} \cup >_{w_{\langle \mathsf{w} \rangle}}$ constitutes a strict total order over the elements of ρ_v and $\rho_{w_{\langle \mathsf{w} \rangle}}$.

From Eqs. A.25, A.32, A.34, and A.36, A.38 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, Eq. A.16 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

[$\langle \mathsf{w} \rangle > 0, \langle \mathsf{b} \rangle > 0, \langle \mathsf{w} \rangle \neq \langle \mathsf{b} \rangle$] In this case, the constructed plan ρ should perform more than one value change of v, with changes of v to w_v and b_v being performed by (a pair of types of) actions prevailed by the value of $w_{\langle \mathsf{w} \rangle}$ and $w_{\langle \mathsf{b} \rangle}$, respectively. From $\alpha < \infty$, we have $\varphi([\![\delta_{\mathsf{w}}(\langle \mathsf{w} \rangle), \delta_{\mathsf{b}}(\langle \mathsf{w} \rangle), \eta]\!], \sigma_{w_{\langle \mathsf{w} \rangle}}) = \varphi([\![\delta_{\mathsf{w}}(\langle \mathsf{b} \rangle), \delta_{\mathsf{b}}(\langle \mathsf{b} \rangle), \eta]\!], \sigma_{w_{\langle \mathsf{b} \rangle}}) = 0$, and this is due to the respective satisfaction of the conditions of cases (2) and (3) in Eq. A.20. Given that, we distinguish between the following four settings[1].

(1) If $\{a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}, a_{\mathsf{b}_v|\mathsf{b}_{w_{\langle \mathsf{b} \rangle}}}\} \subseteq A_v$, then ρ_v is specified according to Eq. A.23, and its action elements are specified as

$$a_v^i = \begin{cases} a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w} \rangle}}}, & i \text{ is odd} \\ a_{\mathsf{b}_v|\mathsf{b}_{w_{\langle \mathsf{b} \rangle}}}, & i \text{ is even} \end{cases}. \tag{A.39}$$

The relation $>_v$ over the action elements of ρ_v is set according to Eq. A.25, the relation $>_{v,w_{\langle \mathsf{w} \rangle}}$ over the action elements of ρ_v and $\rho_{w_{\langle \mathsf{w} \rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{w} \rangle}} = \{a_v^i < a_{w_{\langle \mathsf{w} \rangle}}^1 \mid i \text{ is odd}, a_v^i \in \rho_v, a_{w_{\langle \mathsf{w} \rangle}}^1 \in \rho_{w_{\langle \mathsf{w} \rangle}}\} \tag{A.40}$$

and the relation $>_{v,w_{\langle \mathsf{b} \rangle}}$ over the action elements of ρ_v and $\rho_{w_{\langle \mathsf{b} \rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{b} \rangle}} = \{a_v^i < a_{w_{\langle \mathsf{b} \rangle}}^1 \mid i \text{ is even}, a_v^i \in \rho_v, a_{w_{\langle \mathsf{b} \rangle}}^1 \in \rho_{w_{\langle \mathsf{b} \rangle}}\} \tag{A.41}$$

For all $w \in \mathsf{pred}(v) \setminus \{w_{\langle \mathsf{w} \rangle}, w_{\langle \mathsf{b} \rangle}\}$, we set $>_{v,w} = \emptyset$.

[1] While the details are slightly different, the four settings here are conceptually similar to these in the previously considered case of $\langle \mathsf{w} \rangle > 0, \langle \mathsf{b} \rangle > 0, \langle \mathsf{w} \rangle = \langle \mathsf{b} \rangle$.

(2) Otherwise, if $\{a_{\mathsf{w}_v|\mathsf{w}_{w_{\langle \mathsf{w}\rangle}}}, a_{\mathsf{b}_v|\mathsf{b}_{w_{\langle \mathsf{b}\rangle}}}\} \subseteq A_v$ and $|\sigma_{w_{\langle \mathsf{w}\rangle}}| > 1$, then we have $|\rho_{w_{\langle \mathsf{w}\rangle}}| \geq 1$. Given that, we again set ρ_v according to Eq. A.23, but now with its action elements being set as

$$a_v^i = \begin{cases} a_{\mathsf{w}_v|\mathsf{w}_{w_{\langle \mathsf{w}\rangle}}}, & i \text{ is odd} \\ a_{\mathsf{b}_v|\mathsf{w}_{w_{\langle \mathsf{b}\rangle}}}, & i \text{ is even} \end{cases}. \tag{A.42}$$

The relation $>_v$ is set according to Eq. A.25, $>_{v,w_{\langle \mathsf{w}\rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{w}\rangle}} = \bigcup_{a_v^i \in \rho_v,\, i \text{ is odd}} \{a^1_{w_{\langle \mathsf{w}\rangle}} < a_v^i\} \cup \{a_v^i < a^2_{w_{\langle \mathsf{w}\rangle}} \mid a^2_{w_{\langle \mathsf{w}\rangle}} \in \rho_{w_{\langle \mathsf{w}\rangle}}\} \tag{A.43}$$

and $>_{v,w_{\langle \mathsf{b}\rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{b}\rangle}} = \{a_v^i < a^1_{w_{\langle \mathsf{b}\rangle}} \mid i \text{ is even}, a_v^i \in \rho_v, a^1_{w_{\langle \mathsf{b}\rangle}} \in \rho_{w_{\langle \mathsf{b}\rangle}}\} \tag{A.44}$$

For all $w \in \mathsf{pred}(v) \setminus \{w_{\langle \mathsf{w}\rangle}, w_{\langle \mathsf{b}\rangle}\}$, we set $>_{v,w} = \emptyset$.

(3) Otherwise, if $\{a_{\mathsf{b}_v|\mathsf{w}_{w_{\langle \mathsf{b}\rangle}}}, a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w}\rangle}}}\} \subseteq A_v$, and $|\sigma_{w_{\langle \mathsf{b}\rangle}}| > 1$, then we have $|\rho_{w_{\langle \mathsf{b}\rangle}}| \geq 1$. Given that, we ρ_v is specified according to Eq. A.23, and its action elements are specified as

$$a_v^i = \begin{cases} a_{\mathsf{w}_v|\mathsf{b}_{w_{\langle \mathsf{w}\rangle}}}, & i \text{ is odd} \\ a_{\mathsf{b}_v|\mathsf{w}_{w_{\langle \mathsf{b}\rangle}}}, & i \text{ is even} \end{cases}. \tag{A.45}$$

The relation $>_v$ is set according to Eq. A.25, $>_{v,w_{\langle \mathsf{w}\rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{w}\rangle}} = \{a_v^i < a^1_{w_{\langle \mathsf{w}\rangle}} \mid i \text{ is odd}, a_v^i \in \rho_v, a^1_{w_{\langle \mathsf{w}\rangle}} \in \rho_{w_{\langle \mathsf{w}\rangle}}\} \tag{A.46}$$

and $>_{v,w_{\langle \mathsf{b}\rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{b}\rangle}} = \bigcup_{a_v^i \in \rho_v,\, i \text{ is even}} \{a^1_{w_{\langle \mathsf{b}\rangle}} < a_v^i\} \cup \{a_v^i < a^2_{w_{\langle \mathsf{b}\rangle}} \mid a^2_{w_{\langle \mathsf{b}\rangle}} \in \rho_{w_{\langle \mathsf{b}\rangle}}\} \tag{A.47}$$

For all $w \in \mathsf{pred}(v) \setminus \{w_{\langle \mathsf{w}\rangle}\}$, we set $>_{v,w} = \emptyset$.

(4) Otherwise, if $\{a_{\mathsf{w}_v|\mathsf{w}_{w_{\langle \mathsf{w}\rangle}}}, a_{\mathsf{b}_v|\mathsf{w}_{w_{\langle \mathsf{b}\rangle}}}\} \subseteq A_v$, $|\sigma_{w_{\langle \mathsf{w}\rangle}}| > 1$, and $|\sigma_{w_{\langle \mathsf{b}\rangle}}| > 1$, then we have both $|\rho_{w_{\langle \mathsf{w}\rangle}}| \geq 1$ and $|\rho_{w_{\langle \mathsf{b}\rangle}}| \geq 1$. Given that, we again set ρ_v according to Eq. A.23, and its action elements are specified as

$$a_v^i = \begin{cases} a_{\mathsf{w}_v|\mathsf{w}_{w_{\langle \mathsf{w}\rangle}}}, & i \text{ is odd} \\ a_{\mathsf{b}_v|\mathsf{b}_{w_{\langle \mathsf{b}\rangle}}}, & i \text{ is even} \end{cases}. \tag{A.48}$$

The relation $>_v$ is set according to Eq. A.25, $>_{v,w_{\langle \mathsf{w}\rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{w}\rangle}} = \bigcup_{a_v^i \in \rho_v,\, i \text{ is odd}} \{a^1_{w_{\langle \mathsf{w}\rangle}} < a_v^i\} \cup \{a_v^i < a^2_{w_{\langle \mathsf{w}\rangle}} \mid a^2_{w_{\langle \mathsf{w}\rangle}} \in \rho_{w_{\langle \mathsf{w}\rangle}}\} \tag{A.49}$$

and $>_{v,w_{\langle \mathsf{b}\rangle}}$ is set to

$$>_{v,w_{\langle \mathsf{b}\rangle}} = \bigcup_{a_v^i \in \rho_v,\, i \text{ is even}} \{a^1_{w_{\langle \mathsf{b}\rangle}} < a_v^i\} \cup \{a_v^i < a^2_{w_{\langle \mathsf{b}\rangle}} \mid a^2_{w_{\langle \mathsf{b}\rangle}} \in \rho_{w_{\langle \mathsf{b}\rangle}}\} \tag{A.50}$$

For all $w \in \mathsf{pred}(v) \setminus \{w_{\langle \mathsf{w}\rangle}\}$, we set $>_{v,w} = \emptyset$.

In all the four cases above, both $>_v \cup >_{v,w_{\langle \mathsf{w} \rangle}} \cup >_{w_{\langle \mathsf{w} \rangle}}$ and $>_v \cup >_{v,w_{\langle \mathsf{b} \rangle}} \cup >_{w_{\langle \mathsf{b} \rangle}}$ constitute strict total orders over their respective domains.

From Eqs. A.25, A.40, A.41, A.43, A.44, A.46, A.47, A.49, and A.50 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, Eq. A.16 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

Until now, for each variable $v \in V$, we have specified an action sequence ρ_v and the order $>_v$ over the elements of ρ_v. For each $w \in \mathsf{pred}(v)$, we have specified the order $>_{v,w}$, and proved that all $>_v \cup >_{v,w}$ and $>_w \cup >_{v,w}$ form strict partial orders over their domains, and any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$ and provides to v the value $G[v]$ if the latter is specified. This construction allows us to apply now Theorem 22 on the (considered as sets) sequences ρ_v and orders $>_v$ and $>_{v,w}$, proving that

$$>= \bigcup_{v \in V} (>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w})$$

forms a strict partial order over the union of $\rho_{v_1}, \ldots, \rho_{v_n}$.

Here we also note that the plan extraction step of the algorithm polytree-1-dep-uniform corresponds exactly to the above construction along Eqs. A.23-A.50, providing us in polynomial time with a concrete cost-optimal plan corresponding to the optimal solution for COP_Π.

(II) We now prove that if Π is solvable, then we must have $\alpha < \infty$. Assume to the contrary that this is not the case. Let Π be a solvable $\mathbf{P}(1)$ task, and let (using Theorem 24) ρ be an irreducible, post-unique plan for Π. Given such ρ, let a COP assignment $\overline{x}_\rho$ be defined as follows.

1. For each COP variable x_v, the assignment $\overline{x}_\rho$ provides the value $\sigma_v \in \trianglerighteq^*[\sigma(v)]$ such that $|\sigma_v| = |\rho v \downarrow_|| + 1$.
2. For each COP variable x_v^w, the assignment $\overline{x}_\rho$ provides the value $[\![\delta^w_{\mathsf{w}_v}, \delta^w_{\mathsf{b}_v}, |\sigma_v| - 1]\!]$, where $\delta^w_{\mathsf{w}_v} = 1$ if some action in $\rho\downarrow_v$ preconditioned by the value of w changes the value of v to w_v, and $\delta^w_{\mathsf{w}_v} = 0$, otherwise. $\delta^w_{\mathsf{b}_v}$ is defined similarly to $\delta^w_{\mathsf{w}_v}$, *mutatis mutandis*.

From Eq. A.18-A.22 we then directly have that, for all $v \in V$, $\varphi_{x_v}(\overline{x}_\rho) = |\rho \downarrow_v|$, and for all $w \in \mathsf{pred}(v)$, $\varphi_{x_v^w}(\overline{x}_\rho) = 0$. Therefore, we have

$$\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}_\rho) \;=\; \sum_{v \in V} cost(\rho\downarrow_v) \;=\; cost(\rho),$$

which is what we had to prove. ∎

Theorem 25 *Cost-optimal planning for* $\mathbf{P}(1)$ *with uniform cost actions is tractable.*

Proof: The correctness of the polytree-1-dep-uniform algorithm is given by Lemma 10. We now show that, given a planning task $\Pi \in \mathbf{P}(1)$ with uniform cost actions, the corresponding constraint optimization problem COP_Π can be constructed and solved in time polynomial in the description size of Π.

Let n be the number of state variables in Π. In polytree-1-dep-uniform, we first construct the constraint optimization problem COP_Π over $\Theta(n^2)$ variables $\mathcal{X}$ with domain sizes bounded by

$O(n)$, and $\Theta(n^2)$ functional components $\mathcal{F}$, each defined over at most three COP variables. The construction is linear in the size of the resulting COP, and thus is accomplished in time $O(n^5)$.

Applying then to COP_Π a tree-decomposition along the scopes of the functional components $\mathcal{F}$, we arrive into an equivalent, tree-structured constraint optimization problem over $\Theta(n^2)$ variables with domains of size $O(n^3)$. Such a tree-structured COP can be solved in time $O(xy^2)$ where x is the number of variables and y is an upper bound on the size of a variable's domain (Dechter, 2003). Therefore, solving COP_Π can be done in time $O(n^8)$. As this dominates both the time complexity of constructing COP_Π, and the time complexity of extracting a plan from the optimal solution to COP_Π (see the proof of (I) in Lemma 10), the overall complexity of the algorithm polytree-1-dep-uniform is $O(n^8)$, and therefore polynomial in the description size of Π. ■

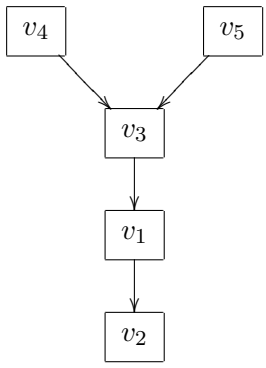
A.3 Cost-Optimal Planning for P(1) with General Action Costs

We now consider cost-optimal planning for $\mathbf{P}(1)$ problems without the constraints on actions to be all of the same cost. While Theorem 24 in Section A.2 shows that any solvable $\mathbf{P}(1)$ planning task Π has at least one post-unique plan, it is possible that no such plan is cost-optimal for Π, and Example 1 below affirms this possibility.

Example 1 *Let $\Pi = \langle V, A, I, G, cost\rangle$ be the $\mathbf{P}(1)$ planning task over variables $V = \{v_1, \ldots, v_5\}$, $I = \{0, 0, 0, 0, 0\}$, $G = \{v_1 = 0, v_2 = 1, v_3 = 1\}$, and actions A as depicted in Figure A.5a. The polytree causal graph of Π is shown in Figure A.5b, and it is easy verify from the table in Figure A.5a that $\Pi \in \mathbf{P}(1)$.*

A	$\mathsf{pre}(a)$					$\mathsf{eff}(a)$					$cost(a)$
	v_1	v_2	v_3	v_4	v_5	v_1	v_2	v_3	v_4	v_5	
a_1	1	0					1				1.0
a_2	0		1			1					1.0
a_3	1		0			0					1.0
a_4			0		0			1			10.0
a_5			1	0				0			1.0
a_6			0	1				1			1.0
a_7				0					1		1.0

(a)



(b)

Figure A.5: Action set and the causal graph for the task in Example 1

Ignoring the non-uniformity of the action costs, this task has a post-unique cost-optimal optimal plan $\rho = \langle a_4 \cdot a_2 \cdot a_1 \cdot a_5 \cdot a_3 \cdot a_4\rangle$. However, considering the action costs as in in the last column in Figure A.5a, then the cost of each optimal plan, such as $\rho' = \langle a_4 \cdot a_2 \cdot a_1 \cdot a_5 \cdot a_3 \cdot a_7 \cdot a_6\rangle$ will be $cost(\rho') = 16 < 24 = cost(\rho)$. Note that the plan ρ' is not post-unique because it changes the value of v_3 to 1 using both actions a_4 and a_6. In fact, any plan for this task will have at least two action instances that change the value of v_3 to 1. However, the cheap such action a_6 cannot be applied twice because it requires $v_4 = 1$, and the only action a_5 that sets $v_3 = 0$ cannot be applied after a_6—

a_5 requires $v_4 = 0$, and there is no action that have this effect. Therefore, any post-unique plan for this task will have to invoke twice the action a_4, and thus will have cost of at least $2 \cdot cost(a_4) = 20$.

Fortunately, here we show that any solvable $\mathbf{P}(1)$ task is guaranteed to have a cost-optimal plan satisfying a certain "relaxation" of action sequence post-uniqueness that still allows us to devise a (more costly than polytree-1-dep-uniform) planning-to-COP scheme for *general* $\mathbf{P}(1)$ problems.

A.3.1 Post-3/2 Plans and P(1) Problems

We now proceed with introducing the notion of post-3/2 property for action sequences that relaxes the post-uniqueness property exploited in the previous section.

Definition 22 *Let $\Pi = \langle V, A, I, G, cost \rangle$ be a* UB *planning task. An action sequence $\varrho \in A^*$ is called* **post-3/2** *if, for each $v \in V$, $a \in \varrho_v$, there exist $\alpha \neq \beta \in \{\mathsf{b}_v, \mathsf{w}_v\}$, a parent $w \in \mathsf{pred}(v)$, $\gamma, \delta \in \{\mathsf{b}_w, \mathsf{w}_w\}$, and $\xi \in \{\mathsf{b}_u, \mathsf{w}_u \mid u \in \mathsf{pred}(v)\}$, such that $a \in \{a_{\alpha|\gamma}, a_{\beta|\delta}, a_{\alpha|\xi}\}$. That is, all the changes of each variable are done using at most three types of actions which are prevailed by at most two parents, and if u is different from w, then different actions prevailed by w perform different value changes of v.*

The (possibly empty) set of all **post-3/2 plans** *for Π is denoted by $\mathcal{P}^{3/2}(\Pi)$ (or simply $\mathcal{P}^{3/2}$, if the identity of Π is clear from the context).*

To illustrate the rather involved definition of post-3/2 plans, consider the following four action sequences of four actions each. The only value changes made by these sequences are changes of variable v with $\mathsf{pred}(v) = \{x, y, z\}$.

- $\langle a_{\mathsf{w}_v|\mathsf{b}_x} \cdot a_{\mathsf{b}_v|\mathsf{w}_x} \cdot a_{\mathsf{w}_v|\mathsf{b}_x} \cdot a_{\mathsf{b}_v|\mathsf{b}_y} \rangle$ is post-3/2 because it uses three types of actions, each prevailed by one of the two parents x, y.
- $\langle a_{\mathsf{w}_v|\mathsf{b}_x} \cdot a_{\mathsf{b}_v|\mathsf{b}_y} \cdot a_{\mathsf{w}_v|\mathsf{b}_x} \cdot a_{\mathsf{b}_v|\mathsf{b}_y} \rangle$ is post-3/2 because it uses two types of actions, each prevailed by one of the two parents x, y.
- $\langle a_{\mathsf{w}_v|\mathsf{b}_x} \cdot a_{\mathsf{b}_v|\mathsf{b}_y} \cdot a_{\mathsf{w}_v|\mathsf{b}_x} \cdot a_{\mathsf{b}_v|\mathsf{b}_z} \rangle$ is *not* post-3/2 because it uses three types of actions, each prevailed by one of the *three parents* x, y, z.
- $\langle a_{\mathsf{w}_v|\mathsf{b}_x} \cdot a_{\mathsf{b}_v|\mathsf{w}_x} \cdot a_{\mathsf{w}_v|\mathsf{b}_y} \cdot a_{\mathsf{b}_v|\mathsf{b}_y} \rangle$ is *not* post-3/2 because it uses *four types of actions*, each prevailed by one of the two parents x, y.

It is not hard to verify that post-3/2 is a relaxation of post-uniqueness—if a plan is post-unique, then it is post-3/2, but not necessarily the other way around. Turns out that, for any $\mathbf{P}(1)$ task Π, this relaxed property is guaranteed to be satisfied by at least one cost-optimal plan for Π.

Theorem 26 *For every solvable $\mathbf{P}(1)$ planning task $\Pi = \langle V, A, I, G, cost \rangle$, the plan set $\mathcal{P}^{3/2}(\Pi)$ contains at least one cost-optimal plan.*

Proof: Given a $\mathbf{P}(1)$ task $\Pi = \langle V, A, I, G, cost \rangle$, and cost-optimal plan $\rho a_1, \ldots, a_m \rangle$ for Π, we construct a sequence of actions ρ^* such that:

- ρ^* is a *post*-3/2 plan for Π,
- $cost(\rho^*) = cost(\rho)$.

In nutshell, first, for each $v \in V$, we map the subsequence $\rho[v] = \langle a_1, \ldots, a_k \rangle$ of ρ into a sequence of actions $\rho_v^* = \langle a_1^*, \ldots, a_k^* \rangle$ that (i) satisfy the *post*-3/2 property, and (ii) $cost(\rho_v^*) \leq cost(\rho[v])$. Then, we merge the constructed sequences $\{\rho_v^*\}_{v \in V}$ into ρ^*, and show that ρ^* is a valid plan for Π. The two properties of ρ^* as required above will then hold immediately because $cost(\rho^*) = cost(\rho)$, and ρ^* being post-3/2 is implied by all its per-variable components ρ_v^* being post-3/2.

For each variable $v \in V$, with $\mathsf{pred}(v) = \emptyset$, we set $\rho_v^* = \rho[v]$ and

$$>_v = \{a_i < a_j \mid a_i, a_j \in \rho[v], i < j\}. \tag{A.51}$$

It is immediate from Eq. A.51 that $>_v$ is a strict total order over the elements of ρ_v^*.

In turn, for each variable $v \in V$ with $\mathsf{pred}(v) \neq \emptyset$, given $\{\sigma_w\}_{w \in \mathsf{pred}(v)}$, such that $|\sigma_w| = |\rho[w]|+1$, let a_i^α be the i'th cheapest action that changes variable v to $\alpha \in \{\mathsf{b}_v, \mathsf{w}_v\}$ and prevailed by some value from $\{\sigma_w\}_{w \in \mathsf{pred}(v)}$. Let us now focus on $a_1^{\mathsf{w}} = a_{\mathsf{w}_v|\gamma}$, $a_2^{\mathsf{w}} = a_{\mathsf{w}_v|\mu}$, $a_1^{\mathsf{b}} = a_{\mathsf{b}_v|\delta}$, $a_2^{\mathsf{b}} = a_{\mathsf{b}_v|\nu}$.[2]

(I) If $\gamma = \delta \in \{\mathsf{b}_w, \mathsf{w}_w\}$, we set

$$a_i^* = \begin{cases} a_{\mathsf{w}_v|\gamma} & i = 2j-1, j \in \mathbb{N} \\ a_{\mathsf{b}_v|\gamma} & \text{otherwise} \end{cases} \tag{A.52}$$

In addition, we construct the following sets of ordering constraints. First, we set a binary relation $>_v$ over the action elements of $\rho_v^* = \langle a_1^*, \ldots, a_k^* \rangle$ to

$$>_v = \{a_i^* < a_j^* \mid a_i^*, a_j^* \in \rho_v^*, i < j\}. \tag{A.53}$$

It is immediate from Eq. A.53 that $>_v$ is a strict total order over the elements of ρ_v^*. Likewise, if $\rho_w^* = \langle a_{j_1}, \ldots, a_{j_l} \rangle$, we set

$$>_{v,w} = \begin{cases} \bigcup_{a_i^* \in \rho_v^*} \{a_i^* < a_{j_1}\}, & \gamma = \mathsf{b}_w \\ \bigcup_{a_i^* \in \rho_v^*} \{a_i^* > a_{j_1}\}, & \gamma = \mathsf{w}_w, l = 1 \\ \bigcup_{a_i^* \in \rho_v^*} \{a_i^* > a_{j_1}\} \cup \{a_i^* < a_{j_2}\}, & \gamma = \mathsf{w}_w, l > 1 \end{cases} \tag{A.54}$$

Finally, the ordering constraints $>_{v,w'}$ for the rest of the parents $w' \in \mathsf{pred}(v) \setminus \{w\}$ are set to empty sets.

For each $w \in \mathsf{pred}(v)$, it is easy to verify that the relation $>_{v,w}$ defined by Eq. A.54 is a strict total order over its domain. Also, from Eqs. A.53 and A.54, we have that, for each $w \in \mathsf{pred}(v)$, $>_v \cup >_{v,w}$ is a strict total order over the union of the elements of ρ_v^* and ρ_w^*.

From Eqs. A.53-A.54 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.52 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

[2] It is possible that some of these actions do not exist, and our case by case analysis in the proof transparently takes this possibility into account. Specifically, if a_1^{w} does not exist, then variable v simply unchangeable, meaning $\rho[v] = \emptyset$. Next, if a_1^{b} does not exist, then v can be changed at most once (from b to w), and this is covered by a subcase of (I). If a_2^{w} does not exist, and a_2^{b} does exist, then only sub-cases (a) of the cases $\{(III), (IV)\}.\{1, 3, 5, 7\}$ are possible. Similarly, if a_2^{b} does not exist, and a_2^{w} does exist, then only sub-cases (b) of the cases $\{(III), (IV)\}.\{1, 3, 5, 7\}$ are possible. Finally, if both a_2^{w} and a_2^{b} do not exist, then cases $\{(III), (IV)\}.\{1, 3, 5, 7\}$ are not possible at all.

(II) If $\gamma \in \{\mathsf{b}_w, \mathsf{w}_w\}$ and $\delta \in \{\mathsf{b}_u, \mathsf{w}_u\}$, such that $w \neq u$, we set

$$a_i^* = \begin{cases} a_{\mathsf{w}_v|\gamma} & i = 2j-1, j \in \mathbb{N} \\ a_{\mathsf{b}_v|\delta} & \text{otherwise} \end{cases} \tag{A.55}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_l \rangle$, and $\rho_u^* = \langle a'_1, \ldots, a'_{l'} \rangle$, we set $>_{v,w}$ according to Eq. A.54 above, and $>_{v,u}$ according to Eq. A.56 below.

$$>_{v,u} = \begin{cases} \bigcup_{a_i^* \in \rho_v^*} \{a_i^* < a'_1\}, & \nu = \mathsf{b}_u \\ \bigcup_{a_i^* \in \rho_v^*} \{a_i^* > a'_1\}, & \nu = \mathsf{w}_u, l' = 1 \\ \bigcup_{a_i^* \in \rho_v^*} \{a_i^* > a'_1\} \cup \{a_i^* < a'_2\}, & \delta = \mathsf{w}_u, l' > 1 \end{cases} \tag{A.56}$$

Finally, here as well, the ordering constraints $>_{v,w'}$ for the rest of the parents $w' \in \mathsf{pred}(v) \setminus \{u, w\}$ are set to empty sets.

The relations $>_v$ here is identical to these in previous case, and relations $>_{v,u}$ and $>_{v,w}$ are effectively identical to the relation $>_{v,w}$ in previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.

From Eqs. A.53, A.54, A.56 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.55 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(III) If $\gamma = \mathsf{b}_w, \delta = \mathsf{w}_w$, we distinguish between a few cases based on σ_w and σ_v.

(1) If $|\rho[v]| = 2y+1$, $|\sigma_w| = 2x$, $|\sigma_w| \leq |\rho[v]|$, then we construct two post-3/2 candidates for ρ_v^*, and then assign ρ_v^* to the cheapest among the two, proving that its cost has to be lower than $cost(\rho[v])$.

(a) *All the changes of v to w_v are done using action a_1^{w}, and then the maximally possible number of changes to b_v are done using action a_1^{b}, with the remaining changes to b_v being done using action a_2^{b}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_1^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, j < x \\ a_2^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, x \leq j \leq y \\ a_1^{\mathsf{w}} & \text{otherwise} \end{cases} \tag{A.57}$$

And the cost in this case is

$$(y+1) \cdot cost(a_1^{\mathsf{w}}) + (x-1) \cdot cost(a_1^{\mathsf{b}}) + (y-x+1) \cdot cost(a_2^{\mathsf{b}}) \tag{A.58}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x-1} \rangle$, and $\rho_u^* = \langle a'_1, \ldots, a'_{l'} \rangle$, we set $>_{v,w}$ according to Eq. A.59, and $>_{v,u}$ according to Eq. A.60.

$$>_{v,w} = \bigcup_{a_i^* \in \rho_v^*, a_j \in \rho_w^*} \{a_i^* < a_j \mid i \leq j < 2x-1\} \cup \{a_i^* < a_{2x-1}\} \cup \{a_j < a_i^* \mid j < i, j < 2x-1\} \tag{A.59}$$

For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set,

$$>_{v,u} = \begin{cases} \bigcup_{a_i^* \in \rho_v^*} \{a_i^* < a_1'\}, & \nu = \mathsf{b}_u \\ \bigcup_{a_i^* \in \rho_v^*} \{a_i^* > a_1'\}, & \nu = \mathsf{w}_u, l' = 1 \\ \bigcup_{a_i^* \in \rho_v^*} \{a_i^* > a_1'\} \cup \{a_i^* < a_2'\}, & \nu = \mathsf{w}_u, l' > 1 \\ \emptyset, & \text{otherwise} \end{cases}. \tag{A.60}$$

It is not hard to verify that the relation $>_{v,w}$ defined by Eq. A.59 is a strict total order over its domain. Suppose to the contrary that for some i, j, both $a_j < a_i^*$ and $a_i^* < a_j$. Then from first inequality we have either $i \leq j < 2x - 1$ or $j = 2x - 1$, and from second we have $j < i, j < 2x - 1$.
The relations $>_v$ and $>_{v,u}$ are effectively identical to these in case (II). Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.
From Eqs. A.53, A.59, and A.60 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.57 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(b) *All the changes of v to b_v are done using action a_1^{b}, and then the maximally possible number of changes to w_v are done using action a_1^{w}, with the remaining changes to w_v being done using action a_2^{w}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_1^{\mathsf{w}} & i = 2j - 1, j \in \mathbb{N}, j \leq x \\ a_2^{\mathsf{w}} & i = 2j - 1, j \in \mathbb{N}, x < j \leq y + 1 \\ a_1^{\mathsf{b}} & \text{otherwise} \end{cases} \tag{A.61}$$

And the cost in this case is

$$x \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}}) \tag{A.62}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x-1} \rangle$, and $\rho_u^* = \langle a_1', \ldots, a_{l'}' \rangle$, we set $>_{v,w}$ according to Eq. A.63, and $>_{v,u}$ according to Eq. A.64.

$$>_{v,w} = \bigcup_{a_i^* \in \rho_v^*, a_j \in \rho_w^*} \{a_i^* < a_j \mid i \leq j\} \cup \{a_j < a_i^* \mid j < i\} \tag{A.63}$$

For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set,

$$>_{v,u} = \begin{cases} \bigcup_{a_i^* \in \rho_v^*} \{a_i^* < a_1'\}, & \mu = \mathsf{b}_u \\ \bigcup_{a_i^* \in \rho_v^*} \{a_i^* > a_1'\}, & \mu = \mathsf{w}_u, l' = 1 \\ \bigcup_{a_i^* \in \rho_v^*} \{a_i^* > a_1'\} \cup \{a_i^* < a_2'\}, & \mu = \mathsf{w}_u, l' > 1 \\ \emptyset, & \text{otherwise} \end{cases}. \tag{A.64}$$

The relation $>_{v,w}$ defined by Eq. A.63 is a strict total order over its domain. The relations $>_v$ and $>_{v,u}$ are effectively identical to these in case (II). Thus, here as

well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.
From Eqs. A.53, A.63, and A.64 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.61 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

Now, for each cost-optimal plan ρ, $\rho[v]$ cannot contain more than $y + x$ actions of both types a_1^{w} and a_1^{b} totally. Suppose to the contrary that $\rho[v]$ contain at least $y + x + 1$ actions of types a_1^{w} and a_1^{b}. Then it contains no more than $y - x$ actions of other types. Let $\mathsf{b}_w \cdot \mathsf{w}_w \cdot \ldots \cdot \mathsf{b}_w$ sequence of $2y + 1$ values of w that support cost-optimal plan for v given that w can change it value any number of times. Then each action of other type will decrease the needed length of this sequence in at most 2. Therefore at most $y - x$ actions of other type will decrease the length in at most $2y - 2x$, and we are left with the sequence of length $\geq 2y + 1 - (2y - 2x) = 2x + 1$. Therefore σ_w cannot support more than $y + x$ actions of types a_1^{w} and a_1^{b}. Now, suppose that in some given cost-optimal plan $\rho[v]$ for v there are α actions of type a_1^{w} and β actions of type a_1^{b}. Then

$$\alpha + \beta \leq y + x \tag{A.65}$$

and

$$cost(\rho_v) \geq \alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) \tag{A.66}$$

For $(A.58) \leq (A.62)$, we have

$$cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \leq cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \tag{A.67}$$

Now suppose to the contrary that the plan in first case is not cost-optimal. Then from Eq. A.66 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) <$$
$$(y + 1) \cdot cost(a_1^{\mathsf{w}}) + (x - 1) \cdot cost(a_1^{\mathsf{b}}) + (y - x + 1) \cdot cost(a_2^{\mathsf{b}})$$

and from it

$$(y + 1 - \alpha) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) < (\beta - x + 1) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) \tag{A.68}$$

From Eq. A.65 we have $y + 1 - \alpha \geq \beta - x + 1$, together with Eq. A.67 contradicting Eq. A.68.
For $(A.62) \leq (A.58)$, we have

$$cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \leq cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \tag{A.69}$$

Now suppose to the contrary that the plan in second case is not cost-optimal. Then from Eq. A.66 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) <$$
$$x \cdot cost(a_1^{\mathsf{w}}) + (y - x + 1) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}})$$

and from it

$$(y-\beta)\cdot(cost(a_2^{\mathsf{b}})-cost(a_1^{\mathsf{b}}))<(\alpha-x)\cdot(cost(a_2^{\mathsf{w}})-cost(a_1^{\mathsf{w}})) \tag{A.70}$$

From Eq. A.65 we have $y-\beta\geq\alpha-x$, together with Eq. A.69 contradicting Eq. A.70.

(2) If $|\rho_v|=2y+1$, $|\sigma_w|=2x$, $|\sigma_w|>|\rho_v|$, then the actions of ρ_v^* are set to

$$a_i^*=\begin{cases}a_{\mathsf{w}_v|\mathsf{b}_w} & i=2j-1, j\in\mathbb{N}\\ a_{\mathsf{b}_v|\mathsf{w}_w} & \text{otherwise}\end{cases} \tag{A.71}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^*=\langle a_1,\ldots,a_{2x-1}\rangle$, we set $>_{v,w}$ according to Eq. A.63 above.
Finally, here as well, the ordering constraints $>_{v,w'}$ for the rest of the parents $w'\in\mathsf{pred}(v)\setminus\{u,w\}$ are set to empty sets.
The relations $>_v$ and $>_{v,w}$ are identical to the previous case. Thus, here as well, we have $>_v\cup>_{v,w}$ forming strict partial order over the union of the elements of ρ_v^* and ρ_w^*.
From Eqs. A.53, A.63 we can now derive that any linearization of $>_v\cup\bigcup_{w\in\mathsf{pred}(v)}>_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\}\cup\mathsf{pred}(v)$. In addition, $|\rho_v^*|=|\rho[v]|$ together with Eq. A.71 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(3) If $|\rho_v|=2y$, $|\sigma_w|=2x$, $|\sigma_w|<|\rho_v|$, then we construct two post-3/2 candidates for ρ_v^*, and then assign ρ_v^* to the cheapest among the two, proving that its cost has to be lower than $cost(\rho[v])$.

(a) *All the changes of v to w_v are done using action a_1^{w}, and then the maximally possible number of changes to b_v are done using action a_1^{b}, with the remaining changes to b_v being done using action a_2^{b}.* For this candidate for ρ_v^*, we set

$$a_i^*=\begin{cases}a_2^{\mathsf{b}} & i=2j, j\in\mathbb{N}, j\leq y-x\\ a_1^{\mathsf{b}} & i=2j, j\in\mathbb{N}, y-x<j\leq y\\ a_1^{\mathsf{w}} & \text{otherwise}\end{cases} \tag{A.72}$$

And the cost in this case is

$$y\cdot cost(a_1^{\mathsf{w}})+x\cdot cost(a_1^{\mathsf{b}})+(y-x)\cdot cost(a_2^{\mathsf{b}}) \tag{A.73}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^*=\langle a_1,\ldots,a_{2x-1}\rangle$, we set $>_{v,w}$ according to Eq. A.74.

$$>_{v,w}=\bigcup_{a_i^*\in\rho_v^*,a_j\in\rho_w^*}\{a_i^*<a_j\mid i\leq 2y-2x+j\}\cup\{a_j<a_i^*\mid i>2y-2x+j\} \tag{A.74}$$

For each $u\in\mathsf{pred}(v)\setminus\{w\}$ we set $>_{v,u}$ according to Eq. A.60. It is easy to verify that the relation $>_{v,w}$ defined by Eq. A.74 is a strict total order over its domain. The relations $>_v$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v\cup>_{v,u}$ and $>_v\cup>_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.

From Eqs. A.53, A.60, and A.74 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.72 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(b) *All the changes of v to b_v are done using action a_1^{b}, and then the maximally possible number of changes to w_v are done using action a_1^{w}, with the remaining changes to w_v being done using action a_2^{w}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_1^{\mathsf{w}} & i = 2j-1, j \in \mathbb{N}, j \leq x \\ a_2^{\mathsf{w}} & i = 2j-1, j \in \mathbb{N}, x < j \leq y \\ a_1^{\mathsf{b}} & \text{otherwise} \end{cases} \tag{A.75}$$

And the cost in this case is

$$x \cdot cost(a_1^{\mathsf{w}}) + (y-x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}}) \tag{A.76}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.63 above.
For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.64.
The relations $>_v$, $>_{v,w}$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.
From Eqs. A.53, A.63, and A.64 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.75 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

Now, for each cost-optimal plan ρ, $\rho[v]$ cannot contain more than $y+x$ actions of both types a_1^{w} and a_1^{b} totally. Suppose to the contrary that $\rho[v]$ contain at least $y+x+1$ actions of types a_1^{w} and a_1^{b}. Then it contains no more than $y-x-1$ actions of other types. Let $\mathsf{b}_w \cdot \mathsf{w}_w \cdot \ldots \cdot \mathsf{w}_w$ sequence of $2y$ values of w that support cost-optimal plan for v given that w can change it value any number of times. Then each action of other type will decrease the needed length of this sequence in at most 2. Therefore at most $y-x-1$ actions of other type will decrease the length in at most $2y-2x-2$, and we are left with the sequence of length $\geq 2y-(2y-2x-2) = 2x+2$. Therefore σ_w cannot support more than $y+x$ actions of types a_1^{w} and a_1^{b}. Now, suppose that in some given cost-optimal plan $\rho[v]$ for v there are α actions of type a_1^{w} and β actions of type a_1^{b}. Then

$$\alpha + \beta \leq y + x \tag{A.77}$$

and

$$cost(\rho_v) \geq \alpha \cdot cost(a_1^{\mathsf{w}}) + (y-\alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y-\beta) \cdot cost(a_2^{\mathsf{b}}) \tag{A.78}$$

For $(A.73) \leq (A.76)$, we have

$$cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \leq cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \tag{A.79}$$

Now suppose to the contrary that the plan in first case is not cost-optimal. Then from Eq. A.78 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y-\alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y-\beta) \cdot cost(a_2^{\mathsf{b}}) < \\ y \cdot cost(a_1^{\mathsf{w}}) + x \cdot cost(a_1^{\mathsf{b}}) + (y-x) \cdot cost(a_2^{\mathsf{b}})$$

and from it

$$(y-\alpha) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) < (\beta - x) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) \tag{A.80}$$

From Eq. A.77 we have $y - \alpha \geq \beta - x$, together with Eq. A.79 contradicting Eq. A.80. For $(A.76) \leq (A.73)$, we have

$$cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \leq cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \tag{A.81}$$

Now suppose to the contrary that the plan in second case is not cost-optimal. Then from Eq. A.78 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y-\alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y-\beta) \cdot cost(a_2^{\mathsf{b}}) < \\ x \cdot cost(a_1^{\mathsf{w}}) + (y-x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}})$$

and from it

$$(y-\beta) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) < (\alpha - x) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) \tag{A.82}$$

From Eq. A.77 we have $y - \beta \geq \alpha - x$, together with Eq. A.81 contradicting Eq. A.82.

(4) If $|\rho_v| = 2y$, $|\sigma_w| = 2x$, $|\sigma_w| \geq |\rho_v|$, then the actions of ρ_v^* are set to

$$a_i^* = \begin{cases} a_{\mathsf{w}_v|\mathsf{b}_w} & i = 2j-1, j \in \mathbb{N} \\ a_{\mathsf{b}_v|\mathsf{w}_w} & \text{otherwise} \end{cases} \tag{A.83}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.63 above. Finally, here as well, the ordering constraints $>_{v,w'}$ for the rest of the parents $w' \in \mathsf{pred}(v) \setminus \{u, w\}$ are set to empty sets. The relations $>_v$ and $>_{v,w}$ are identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,w}$ forming strict partial order over the union of the elements of ρ_v^* and ρ_w^*.

From Eqs. A.53, A.63 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.83 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(5) If $|\rho_v| = 2y+1$, $|\sigma_w| = 2x+1$, $|\sigma_w| < |\rho_v|$, then we construct two post-3/2 candidates for ρ_v^*, and then assign ρ_v^* to the cheapest among the two, proving that its cost has to be lower than $cost(\rho[v])$.

(a) *All the changes of v to w_v are done using action a_1^{w}, and then the maximally possible number of changes to b_v are done using action a_1^{b}, with the remaining changes to b_v being done using action a_2^{b}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_2^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, j \leq y - x \\ a_1^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, y - x < j \leq y \\ a_1^{\mathsf{w}} & \text{otherwise} \end{cases} \tag{A.84}$$

And the cost in this case is

$$(y+1) \cdot cost(a_1^{\mathsf{w}}) + x \cdot cost(a_1^{\mathsf{b}}) + (y-x) \cdot cost(a_2^{\mathsf{b}}) \tag{A.85}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x} \rangle$, we set $>_{v,w}$ according to Eq. A.74 above. For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.60 above. The relations $>_v$, $>_{v,w}$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.
From Eqs. A.53, A.60, and A.74 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.84 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(b) *All the changes of v to b_v are done using action a_1^{b}, and then the maximally possible number of changes to w_v are done using action a_1^{w}, with the remaining changes to w_v being done using action a_2^{w}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_1^{\mathsf{w}} & i = 2j-1, j \in \mathbb{N}, j \leq x \text{ or } j = y+1 \\ a_2^{\mathsf{w}} & i = 2j-1, j \in \mathbb{N}, x < j \leq y \\ a_1^{\mathsf{b}} & \text{otherwise} \end{cases} \tag{A.86}$$

And the cost in this case is

$$(x+1) \cdot cost(a_1^{\mathsf{w}}) + (y-x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}}) \tag{A.87}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x} \rangle$, we set $>_{v,w}$ according to Eq. A.88.

$$>_{v,w} = \bigcup_{a_i^* \in \rho_v^*, a_j \in \rho_w^*} \begin{array}{c} \{a_i^* < a_j \mid i \leq j < 2x\} \cup \{a_i^* < a_{2x} \mid i \leq 2y\} \\ \cup \{a_j < a_i^* \mid j < i, j < 2x\} \cup \{a_{2x} < a_{2y+1}^*\} \end{array} \tag{A.88}$$

For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.64 above.
It is easy to verify that the relation $>_{v,w}$ defined by Eq. A.88 is a strict total order over its domain. The relations $>_v$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.
From Eqs. A.53, A.64, and A.88 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.86 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

Now, for each cost-optimal plan ρ, $\rho[v]$ cannot contain more than $y + x + 1$ actions of both types a_1^{w} and a_1^{b} totally. Suppose to the contrary that $\rho[v]$ contain at least $y+x+2$ actions of types a_1^{w} and a_1^{b}. Then it contains no more than $y - x - 1$ actions of other types. Let $\mathsf{b}_w \cdot \mathsf{w}_w \cdot \ldots \cdot \mathsf{b}_w$ sequence of $2y + 1$ values of w that support cost-optimal plan for v given that w can change it value any number of times. Then each action of other type will decrease the needed length of this sequence in at most 2. Therefore at most $y - x - 1$ actions of other type will decrease the length in at most $2y - 2x - 2$, and we are left with the sequence of length $\geq 2y + 1 - (2y - 2x - 2) = 2x + 3$. Therefore σ_w cannot support more than $y + x + 1$ actions of types a_1^{w} and a_1^{b}. Now, suppose that in some given cost-optimal plan $\rho[v]$ for v there are α actions of type a_1^{w} and β actions of type a_1^{b}. Then

$$\alpha + \beta \leq y + x + 1 \tag{A.89}$$

and

$$cost(\rho_v) \geq \alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) \tag{A.90}$$

For $(A.85) \leq (A.87)$, we have

$$cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \leq cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \tag{A.91}$$

Now suppose to the contrary that the plan in first case is not cost-optimal. Then from Eq. A.90 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) <$$
$$(y + 1) \cdot cost(a_1^{\mathsf{w}}) + x \cdot cost(a_1^{\mathsf{b}}) + (y - x) \cdot cost(a_2^{\mathsf{b}})$$

and from it

$$(y + 1 - \alpha) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) < (\beta - x) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) \tag{A.92}$$

From Eq. A.89 we have $y+1-\alpha \geq \beta-x$, together with Eq. A.91 contradicting Eq. A.92. For $(A.87) \leq (A.85)$, we have

$$cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \leq cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \tag{A.93}$$

Now suppose to the contrary that the plan in second case is not cost-optimal. Then from Eq. A.90 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) <$$
$$(x + 1 \cdot cost(a_1^{\mathsf{w}}) + (y - x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}})$$

and from it

$$(y - \beta) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) < (\alpha - x - 1) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) \tag{A.94}$$

From Eq. A.89 we have $y-\beta \geq \alpha-x-1$, together with Eq. A.93 contradicting Eq. A.94.

(6) If $|\rho_v| = 2y+1$, $|\sigma_w| = 2x+1$, $|\sigma_w| \geq |\rho_v|$, the actions of ρ_v^* are set to

$$a_i^* = \begin{cases} a_{\mathsf{w}_v|\mathsf{b}_w} & i = 2j-1, j \in \mathbb{N} \\ a_{\mathsf{b}_v|\mathsf{w}_w} & \text{otherwise} \end{cases} \tag{A.95}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x} \rangle$, we set $>_{v,w}$ according to Eq. A.63 above. Finally, here as well, the ordering constraints $>_{v,w'}$ for the rest of the parents $w' \in \mathsf{pred}(v) \setminus \{u, w\}$ are set to empty sets. The relations $>_v$ and $>_{v,w}$ are identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,w}$ forming strict partial order over the union of the elements of ρ_v^* and ρ_w^*.

From Eqs. A.53, A.63 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.95 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(7) If $|\rho_v| = 2y$, $|\sigma_w| = 2x+1$, $|\sigma_w| \leq |\rho_v|$, then we construct two post-3/2 candidates for ρ_v^*, and then assign ρ_v^* to the cheapest among the two, proving that its cost has to be lower than $cost(\rho[v])$.

(a) *All the changes of v to w_v are done using action a_1^{w}, and then the maximally possible number of changes to b_v are done using action a_1^{b}, with the remaining changes to b_v being done using action a_2^{b}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_2^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, j \leq y - x \\ a_1^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, y - x < j \leq y \\ a_1^{\mathsf{w}} & \text{otherwise} \end{cases} \tag{A.96}$$

And the cost in this case is

$$y \cdot cost(a_1^{\mathsf{w}}) + x \cdot cost(a_1^{\mathsf{b}}) + (y - x) \cdot cost(a_2^{\mathsf{b}}) \tag{A.97}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x} \rangle$, we set $>_{v,w}$ according to Eq. A.74 above. For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.60 above. The relations $>_v$, $>_{v,w}$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.

From Eqs. A.53, A.60, and A.74 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.96 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(b) *All the changes of v to b_v are done using action a_1^{b}, and then the maximally possible number of changes to w_v are done using action a_1^{w}, with the remaining changes to w_v being done using action a_2^{w}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_1^{\mathsf{w}} & i = 2j-1, j \in \mathbb{N}, j \leq x \\ a_2^{\mathsf{w}} & i = 2j-1, j \in \mathbb{N}, x < j \leq y \\ a_1^{\mathsf{b}} & \text{otherwise} \end{cases} \tag{A.98}$$

And the cost in this case is

$$x \cdot cost(a_1^{\mathsf{w}}) + (y - x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}}) \tag{A.99}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x} \rangle$, we set $>_{v,w}$ according to Eq. A.100.

$$>_{v,w} = \bigcup_{a_i^* \in \rho_v^*, a_j \in \rho_w^*} \{a_i^* < a_j \mid i \leq j < 2x\} \cup \{a_j < a_i^* \mid j < i, j < 2x\} \cup \{a_i^* < a_{2x}\} \tag{A.100}$$

For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.64 above.
It is easy to verify that the relation $>_{v,w}$ defined by Eq. A.100 is a strict total order over its domain. The relations $>_v$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.
From Eqs. A.53, A.64, and A.100 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.98 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

Now, for each cost-optimal plan ρ, $\rho[v]$ cannot contain more than $y + x$ actions of both types a_1^{w} and a_1^{b} totally. Suppose to the contrary that $\rho[v]$ contain at least $y + x + 1$ actions of types a_1^{w} and a_1^{b}. Then it contains no more than $y - x - 1$ actions of other types. Let $\mathsf{b}_w \cdot \mathsf{w}_w \cdot \ldots \cdot \mathsf{w}_w$ sequence of $2y$ values of w that support cost-optimal plan for v given that w can change it value any number of times. Then each action of other type will decrease the needed length of this sequence in at most 2. Therefore at most $y - x - 1$ actions of other type will decrease the length in at most $2y - 2x - 2$, and we are left with the sequence of length $\geq 2y - (2y - 2x - 2) = 2x + 2$. Therefore σ_w cannot support more than $y + x$ actions of types a_1^{w} and a_1^{b}. Now, suppose that in some given cost-optimal plan $\rho[v]$ for v there are α actions of type a_1^{w} and β actions of type a_1^{b}. Then

$$\alpha + \beta \leq y + x \tag{A.101}$$

and

$$cost(\rho_v) \geq \alpha \cdot cost(a_1^{\mathsf{w}}) + (y - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) \tag{A.102}$$

For $(A.97) \leq (A.99)$, we have

$$cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \leq cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \tag{A.103}$$

Now suppose to the contrary that the plan in first case is not cost-optimal. Then from Eq. A.102 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) <$$
$$y \cdot cost(a_1^{\mathsf{w}}) + x \cdot cost(a_1^{\mathsf{b}}) + (y - x) \cdot cost(a_2^{\mathsf{b}})$$

and from it

$$(y - \alpha) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) < (\beta - x) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) \tag{A.104}$$

From Eq. A.101 we have $y-\alpha \geq \beta - x$, together with Eq. A.103 contradicting Eq. A.104.
For $(A.99) \leq (A.97)$, we have

$$cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \leq cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \tag{A.105}$$

Now suppose to the contrary that the plan in second case is not cost-optimal. Then from Eq. A.102 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y-\alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y-\beta) \cdot cost(a_2^{\mathsf{b}}) < \\ x \cdot cost(a_1^{\mathsf{w}}) + (y-x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}})$$

and from it

$$(y-\beta) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) < (\alpha - x) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) \tag{A.106}$$

From Eq. A.101 we have $y-\beta \geq \alpha - x$, together with Eq. A.105 contradicting Eq. A.106.

(8) If $|\rho_v| = 2y$, $|\sigma_w| = 2x+1$, $|\sigma_w| > |\rho_v|$, then the actions of ρ_v^* are set to

$$a_i^* = \begin{cases} a_{\mathsf{w}_v|\mathsf{b}_w} & i = 2j-1, j \in \mathbb{N} \\ a_{\mathsf{b}_v|\mathsf{w}_w} & \text{otherwise} \end{cases} \tag{A.107}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x} \rangle$, we set $>_{v,w}$ according to Eq. A.63 above. Finally, here as well, the ordering constraints $>_{v,w'}$ for the rest of the parents $w' \in \mathsf{pred}(v) \setminus \{u, w\}$ are set to empty sets. The relations $>_v$ and $>_{v,w}$ are identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,w}$ forming strict partial order over the union of the elements of ρ_v^* and ρ_w^*.
From Eqs. A.53, A.63 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.107 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(IV) If $\gamma = \mathsf{w}_w, \delta = \mathsf{b}_w$, we distinguish between a few cases based on σ_w and σ_v.

(1) If $|\rho_v| = 2y+1$, $|\sigma_w| = 2x$, $|\sigma_w| \leq |\rho_v|$, then we construct two post-3/2 candidates for ρ_v^*, and then assign ρ_v^* to the cheapest among the two, proving that its cost has to be lower than $cost(\rho[v])$.

(a) *All the changes of v to w_v are done using action a_1^{w}, and then the maximally possible number of changes to b_v are done using action a_1^{b}, with the remaining changes to b_v being done using action a_2^{b}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_2^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, j \leq y-x+1 \\ a_1^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, y-x+1 < j \leq y \\ a_1^{\mathsf{w}} & \text{otherwise} \end{cases} \tag{A.108}$$

And the cost in this case is

$$(y+1) \cdot cost(a_1^{\mathsf{w}}) + (x-1) \cdot cost(a_1^{\mathsf{b}}) + (y-x+1) \cdot cost(a_2^{\mathsf{b}}) \tag{A.109}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho^*_w = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.110.

$$>_{v,w} = \bigcup_{a^*_i \in \rho^*_v, a_j \in \rho^*_w} \{a^*_i < a_j \mid i < j\} \cup \{a_j < a^*_i \mid j \leq i\} \tag{A.110}$$

For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.60.
It is easy to verify that the relation $>_{v,w}$ defined by Eq. A.110 is a strict total order over its domain. The relations $>_v$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ^*_v and ρ^*_u, and ρ^*_v and ρ^*_w, respectively.
From Eqs. A.53, A.60, and A.110 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho^*_v| = |\rho[v]|$ together with Eq. A.108 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(b) *All the changes of v to b_v are done using action a^{b}_1, and then the maximally possible number of changes to w_v are done using action a^{w}_1, with the remaining changes to w_v being done using action a^{w}_2.* For this candidate for ρ^*_v, we set

$$a^*_i = \begin{cases} a^{\mathsf{w}}_2 & i = 2j-1, j \in \mathbb{N}, j \leq y - x + 1 \\ a^{\mathsf{w}}_1 & i = 2j-1, j \in \mathbb{N}, y - x + 1 < j \leq y + 1 \\ a^{\mathsf{b}}_1 & \text{otherwise} \end{cases} \tag{A.111}$$

And the cost in this case is

$$x \cdot cost(a^{\mathsf{w}}_1) + (y + 1 - x) \cdot cost(a^{\mathsf{w}}_2) + y \cdot cost(a^{\mathsf{b}}_1) \tag{A.112}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho^*_w = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.113.

$$>_{v,w} = \bigcup_{a^*_i \in \rho^*_v, a_j \in \rho^*_w} \{a^*_i < a_j \mid i \leq 2y - 2x + 1 + j\} \cup \{a_j < a^*_i \mid i > 2y - 2x + 1 + j\} \tag{A.113}$$

For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.64.
It is easy to verify that the relation $>_{v,w}$ defined by Eq. A.113 is a strict total order over its domain. The relations $>_v$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ^*_v and ρ^*_u, and ρ^*_v and ρ^*_w, respectively.
From Eqs. A.53, A.64, and A.113 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho^*_v| = |\rho[v]|$ together with Eq. A.111 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

Now, for each cost-optimal plan ρ, $\rho[v]$ cannot contain more than $y + x$ actions of both types a^{w}_1 and a^{b}_1 totally. Suppose to the contrary that $\rho[v]$ contain at least $y + x + 1$ actions of types a^{w}_1 and a^{b}_1. Then it contains no more than $y - x$ actions of other types. Let $\mathsf{w}_w \cdot \mathsf{w}_b \cdot \ldots \cdot \mathsf{w}_w$ sequence of $2y + 1$ values of w that support cost-optimal plan for v

given that w can change it value any number of times. Then each action of other type will decrease the needed length of this sequence in at most 2. Therefore at most $y - x$ actions of other type will decrease the length in at most $2y - 2x$, and we are left with the sequence of length $\geq 2y + 1 - (2y - 2x) = 2x + 1$. Therefore σ_w cannot support more than $y + x$ actions of types a_1^{w} and a_1^{b}. Now, suppose that in some given cost-optimal plan $\rho[v]$ for v there are α actions of type a_1^{w} and β actions of type a_1^{b}. Then

$$\alpha + \beta \leq y + x \tag{A.114}$$

and

$$cost(\rho_v) \geq \alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) \tag{A.115}$$

For $(A.109) \leq (A.112)$, we have

$$cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \leq cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \tag{A.116}$$

Now suppose to the contrary that the plan in first case is not cost-optimal. Then from Eq. A.115 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) <$$
$$(y + 1) \cdot cost(a_1^{\mathsf{w}}) + (x - 1) \cdot cost(a_1^{\mathsf{b}}) + (y - x + 1) \cdot cost(a_2^{\mathsf{b}})$$

and from it

$$(y + 1 - \alpha) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) < (\beta - x + 1) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) \tag{A.117}$$

From Eq. A.114 we have $y + 1 - \alpha \geq \beta - x + 1$, together with Eq. A.116 contradicting Eq. A.117.
For $(A.112) \leq (A.109)$, we have

$$cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \leq cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \tag{A.118}$$

Now suppose to the contrary that the plan in second case is not cost-optimal. Then from Eq. A.115 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) <$$
$$x \cdot cost(a_1^{\mathsf{w}}) + (y - x + 1) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}})$$

and from it

$$(y - \beta) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) < (\alpha - x) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) \tag{A.119}$$

From Eq. A.114 we have $y - \beta \geq \alpha - x$, together with Eq. A.118 contradicting Eq. A.119.

(2) If $|\rho_v| = 2y + 1$, $|\sigma_w| = 2x$, $|\sigma_w| > |\rho_v|$, then the actions of ρ_v^* are set to

$$a_i^* = \begin{cases} a_{\mathsf{w}_v|\mathsf{w}_w} & i = 2j - 1, j \in \mathbb{N} \\ a_{\mathsf{b}_v|\mathsf{b}_w} & \text{otherwise} \end{cases} \tag{A.120}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho^*_w = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.110 above.

Finally, here as well, the ordering constraints $>_{v,w'}$ for the rest of the parents $w' \in \mathsf{pred}(v) \setminus \{u, w\}$ are set to empty sets.

The relations $>_v$ and $>_{v,w}$ are identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,w}$ forming strict partial order over the union of the elements of ρ^*_v and ρ^*_w.

From Eqs. A.53, A.110 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho^*_v| = |\rho[v]|$ together with Eq. A.120 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(3) If $|\rho_v| = 2y$, $|\sigma_w| = 2x$, $|\sigma_w| \leq |\rho_v| + 1$, then we construct two post-3/2 candidates for ρ^*_v, and then assign ρ^*_v to the cheapest among the two, proving that its cost has to be lower than $cost(\rho[v])$.

(a) *All the changes of v to w_v are done using action a^{w}_1, and then the maximally possible number of changes to b_v are done using action a^{b}_1, with the remaining changes to b_v being done using action a^{b}_2.* For this candidate for ρ^*_v, we set

$$a^*_i = \begin{cases} a^{\mathsf{b}}_1 & i = 2j, j \in \mathbb{N}, j < x \\ a^{\mathsf{b}}_2 & i = 2j, j \in \mathbb{N}, x \leq j \leq y \\ a^{\mathsf{w}}_1 & \text{otherwise} \end{cases} \tag{A.121}$$

And the cost in this case is

$$y \cdot cost(a^{\mathsf{w}}_1) + (x-1) \cdot cost(a^{\mathsf{b}}_1) + (y-x+1) \cdot cost(a^{\mathsf{b}}_2) \tag{A.122}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho^*_w = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.110 above.

For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.60.

The relations $>_v$, $>_{v,w}$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ^*_v and ρ^*_u, and ρ^*_v and ρ^*_w, respectively.

From Eqs. A.53, A.60, and A.110 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho^*_v| = |\rho[v]|$ together with Eq. A.121 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(b) *All the changes of v to b_v are done using action a^{b}_1, and then the maximally possible number of changes to w_v are done using action a^{w}_1, with the remaining changes to w_v being done using action a^{w}_2.* For this candidate for ρ^*_v, we set

$$a^*_i = \begin{cases} a^{\mathsf{w}}_2 & i = 2j-1, j \in \mathbb{N}, j \leq y-x+1 \\ a^{\mathsf{w}}_1 & i = 2j-1, j \in \mathbb{N}, y-x+1 < j \leq y \\ a^{\mathsf{b}}_1 & \text{otherwise} \end{cases} \tag{A.123}$$

And the cost in this case is

$$(x-1) \cdot cost(a^{\mathsf{w}}_1) + (y-x+1) \cdot cost(a^{\mathsf{w}}_2) + y \cdot cost(a^{\mathsf{b}}_1) \tag{A.124}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho^*_w = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.113 above.
For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.64.
The relations $>_v$, $>_{v,w}$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ^*_v and ρ^*_u, and ρ^*_v and ρ^*_w, respectively.
From Eqs. A.53, A.64, and A.113 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho^*_v| = |\rho[v]|$ together with Eq. A.123 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

Now, for each cost-optimal plan ρ, $\rho[v]$ cannot contain more than $y+x-1$ actions of both types a^{w}_1 and a^{b}_1 totally. Suppose to the contrary that $\rho[v]$ contain at least $y+x$ actions of types a^{w}_1 and a^{b}_1. Then it contains no more than $y-x$ actions of other types. Let $\mathsf{w}_w \cdot \mathsf{b}_w \cdot \ldots \cdot \mathsf{b}_w$ sequence of $2y$ values of w that support cost-optimal plan for v given that w can change it value any number of times. Then each action of other type will decrease the needed length of this sequence in at most 2. Therefore at most $y-x$ actions of other type will decrease the length in at most $2y-2x$, and we are left with the sequence of length $\geq 2y-(2y-2x) = 2x$, which have to be a subsequence of σ_w, contradicting with the fact that σ_w is of the same or smaller size and starts with a different character. Therefore σ_w cannot support more than $y+x$ actions of types a^{w}_1 and a^{b}_1. Now, suppose that in some given cost-optimal plan $\rho[v]$ for v there are α actions of type a^{w}_1 and β actions of type a^{b}_1. Then

$$\alpha + \beta \leq y + x \tag{A.125}$$

and

$$cost(\rho_v) \geq \alpha \cdot cost(a^{\mathsf{w}}_1) + (y-\alpha) \cdot cost(a^{\mathsf{w}}_2) + \beta \cdot cost(a^{\mathsf{b}}_1) + (y-\beta) \cdot cost(a^{\mathsf{b}}_2) \tag{A.126}$$

For $(A.122) \leq (A.124)$, we have

$$cost(a^{\mathsf{b}}_2) - cost(a^{\mathsf{b}}_1) \leq cost(a^{\mathsf{w}}_2) - cost(a^{\mathsf{w}}_1) \tag{A.127}$$

Now suppose to the contrary that the plan in first case is not cost-optimal. Then from Eq. A.126 we have

$$\begin{aligned} &\alpha \cdot cost(a^{\mathsf{w}}_1) + (y-\alpha) \cdot cost(a^{\mathsf{w}}_2) + \beta \cdot cost(a^{\mathsf{b}}_1) + (y-\beta) \cdot cost(a^{\mathsf{b}}_2) < \\ &y \cdot cost(a^{\mathsf{w}}_1) + x \cdot cost(a^{\mathsf{b}}_1) + (y-x) \cdot cost(a^{\mathsf{b}}_2) \end{aligned}$$

and from it

$$(y-\alpha) \cdot (cost(a^{\mathsf{w}}_2) - cost(a^{\mathsf{w}}_1)) < (\beta - x) \cdot (cost(a^{\mathsf{b}}_2) - cost(a^{\mathsf{b}}_1)) \tag{A.128}$$

From Eq. A.125 we have $y-\alpha \geq \beta - x$, together with Eq. A.127 contradicting Eq. A.128.
For $(A.124) \leq (A.122)$, we have

$$cost(a^{\mathsf{w}}_2) - cost(a^{\mathsf{w}}_1) \leq cost(a^{\mathsf{b}}_2) - cost(a^{\mathsf{b}}_1) \tag{A.129}$$

Now suppose to the contrary that the plan in second case is not cost-optimal. Then from Eq. A.126 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y-\alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y-\beta) \cdot cost(a_2^{\mathsf{b}}) < \\ x \cdot cost(a_1^{\mathsf{w}}) + (y-x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}})$$

and from it

$$(y-\beta) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) < (\alpha - x) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) \tag{A.130}$$

From Eq. A.125 we have $y-\beta \geq \alpha - x$, together with Eq. A.129 contradicting Eq. A.130.

(4) If $|\rho_v| = 2y$, $|\sigma_w| = 2x$, $|\sigma_w| > |\rho_v| + 1$, then the actions of ρ_v^* are set to

$$a_i^* = \begin{cases} a_{\mathsf{w}_v|\mathsf{w}_w} & i = 2j-1, j \in \mathbb{N} \\ a_{\mathsf{b}_v|\mathsf{b}_w} & \text{otherwise} \end{cases} \tag{A.131}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.110 above.

Finally, here as well, the ordering constraints $>_{v,w'}$ for the rest of the parents $w' \in \mathsf{pred}(v) \setminus \{u, w\}$ are set to empty sets.

The relations $>_v$ and $>_{v,w}$ are identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,w}$ forming strict partial order over the union of the elements of ρ_v^* and ρ_w^*.

From Eqs. A.53, A.110 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.131 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(5) If $|\rho_v| = 2y+1$, $|\sigma_w| = 2x+1$, $|\sigma_w| \leq |\rho_v|+1$, then we construct two post-3/2 candidates for ρ_v^*, and then assign ρ_v^* to the cheapest among the two, proving that its cost has to be lower than $cost(\rho[v])$.

(a) *All the changes of v to w_v are done using action a_1^{w}, and then the maximally possible number of changes to b_v are done using action a_1^{b}, with the remaining changes to b_v being done using action a_2^{b}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_1^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, j < x \\ a_2^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, x \leq j \leq y \\ a_1^{\mathsf{w}} & \text{otherwise} \end{cases} \tag{A.132}$$

And the cost in this case is

$$(y+1) \cdot cost(a_1^{\mathsf{w}}) + (x-1) \cdot cost(a_1^{\mathsf{b}}) + (y-x+1) \cdot cost(a_2^{\mathsf{b}}) \tag{A.133}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x} \rangle$, we set $>_{v,w}$ according to Eq. A.134.

$$>_{v,w} = \bigcup_{a_i^* \in \rho_v^*, a_j \in \rho_w^*} \{a_i^* < a_j \mid i < j < 2x\} \cup \{a_j < a_i^* \mid j \leq i, j < 2x\} \cup \{a_i^* < a_{2x}\} \tag{A.134}$$

For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.60.
It is easy to verify that the relation $>_{v,w}$ defined by Eq. A.134 is a strict total order over its domain. The relations $>_v$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.
From Eqs. A.53, A.60, and A.134 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.132 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(b) *All the changes of v to b_v are done using action a_1^{b}, and then the maximally possible number of changes to w_v are done using action a_1^{w}, with the remaining changes to w_v being done using action a_2^{w}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_2^{\mathsf{w}} & i = 2j-1, j \in \mathbb{N}, j \leq y - x \text{ or } j = y+1 \\ a_1^{\mathsf{w}} & i = 2j-1, j \in \mathbb{N}, y - x < j \leq y \\ a_1^{\mathsf{b}} & \text{otherwise} \end{cases} \tag{A.135}$$

And the cost in this case is

$$x \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}}) \tag{A.136}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.137.

$$>_{v,w} = \bigcup_{a_i^* \in \rho_v^*, a_j \in \rho_w^*} \{a_i^* < a_j \mid i < 2y - 2x + j\} \cup \{a_j < a_i^* \mid i \geq 2y - 2x + j\} \tag{A.137}$$

For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.64.
It is easy to verify that the relation $>_{v,w}$ defined by Eq. A.137 is a strict total order over its domain. The relations $>_v$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.
From Eqs. A.53, A.64, and A.137 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.135 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

Now, for each cost-optimal plan ρ, $\rho[v]$ cannot contain more than $y + x + 1$ actions of both types a_1^{w} and a_1^{b} totally. Suppose to the contrary that $\rho[v]$ contain at least $y + x + 2$ actions of types a_1^{w} and a_1^{b}. Then it contains no more than $y - x - 1$ actions of other types. Let $\mathsf{w}_w \cdot \mathsf{b}_w \cdot \ldots \cdot \mathsf{w}_w$ sequence of $2y + 1$ values of w that support cost-optimal plan for v given that w can change it value any number of times. Then each action of other type will decrease the needed length of this sequence in at most 2. Therefore at most $y - x - 1$ actions of other type will decrease the length in at most $2y - 2x - 2$, and we are left with the sequence of length $\geq 2y + 1 - (2y - 2x - 2) = 2x + 3$. Therefore σ_w cannot support more than $y + x + 1$ actions of types a_1^{w} and a_1^{b}. Now, suppose that in

some given cost-optimal plan $\rho[v]$ for v there are α actions of type a_1^{w} and β actions of type a_1^{b}. Then

$$\alpha + \beta \leq y + x + 1 \tag{A.138}$$

and

$$cost(\rho_v) \geq \alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) \tag{A.139}$$

For $(A.133) \leq (A.136)$, we have

$$cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \leq cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \tag{A.140}$$

Now suppose to the contrary that the plan in first case is not cost-optimal. Then from Eq. A.139 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) <$$
$$(y + 1) \cdot cost(a_1^{\mathsf{w}}) + x \cdot cost(a_1^{\mathsf{b}}) + (y - x) \cdot cost(a_2^{\mathsf{b}})$$

and from it

$$(y + 1 - \alpha) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) < (\beta - x) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) \tag{A.141}$$

From Eq. A.138 we have $y + 1 - \alpha \geq \beta - x$, together with Eq. A.140 contradicting Eq. A.141.
For $(A.136) \leq (A.133)$, we have

$$cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}}) \leq cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}}) \tag{A.142}$$

Now suppose to the contrary that the plan in second case is not cost-optimal. Then from Eq. A.139 we have

$$\alpha \cdot cost(a_1^{\mathsf{w}}) + (y + 1 - \alpha) \cdot cost(a_2^{\mathsf{w}}) + \beta \cdot cost(a_1^{\mathsf{b}}) + (y - \beta) \cdot cost(a_2^{\mathsf{b}}) <$$
$$(x + 1 \cdot cost(a_1^{\mathsf{w}}) + (y - x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}})$$

and from it

$$(y - \beta) \cdot (cost(a_2^{\mathsf{b}}) - cost(a_1^{\mathsf{b}})) < (\alpha - x - 1) \cdot (cost(a_2^{\mathsf{w}}) - cost(a_1^{\mathsf{w}})) \tag{A.143}$$

From Eq. A.138 we have $y - \beta \geq \alpha - x - 1$, together with Eq. A.142 contradicting Eq. A.143.

(6) If $|\rho_v| = 2y + 1$, $|\sigma_w| = 2x + 1$, $|\sigma_w| > |\rho_v| + 1$, then the actions of ρ_v^* are set to

$$a_i^* = \begin{cases} a_{\mathsf{w}_v|\mathsf{w}_w} & i = 2j - 1, j \in \mathbb{N} \\ a_{\mathsf{b}_v|\mathsf{b}_w} & \text{otherwise} \end{cases} \tag{A.144}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.110 above.
Finally, here as well, the ordering constraints $>_{v,w'}$ for the rest of the parents $w' \in \mathsf{pred}(v) \setminus \{u, w\}$ are set to empty sets.

The relations $>_v$ and $>_{v,w}$ are identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,w}$ forming strict partial order over the union of the elements of ρ_v^* and ρ_w^*.
From Eqs. A.53, A.110 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.144 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(7) If $|\rho_v| = 2y$, $|\sigma_w| = 2x + 1$, $|\sigma_w| \leq |\rho_v|$, then we construct two post-3/2 candidates for ρ_v^*, and then assign ρ_v^* to the cheapest among the two, proving that its cost has to be lower than $cost(\rho[v])$.

(a) *All the changes of v to w_v are done using action a_1^{w}, and then the maximally possible number of changes to b_v are done using action a_1^{b}, with the remaining changes to b_v being done using action a_2^{b}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_2^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, j \leq y - x \\ a_1^{\mathsf{b}} & i = 2j, j \in \mathbb{N}, y - x < j \leq y \\ a_1^{\mathsf{w}} & \text{otherwise} \end{cases} \tag{A.145}$$

And the cost in this case is

$$y \cdot cost(a_1^{\mathsf{w}}) + x \cdot cost(a_1^{\mathsf{b}}) + (y - x) \cdot cost(a_2^{\mathsf{b}}) \tag{A.146}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^* = \langle a_1, \ldots, a_{2x} \rangle$, we set $>_{v,w}$ according to Eq. A.147.

$$>_{v,w} = \bigcup_{a_i^* \in \rho_v^*, a_j \in \rho_w^*} \{a_i^* < a_j \mid i < 2y - 2x + j, j > 1\} \cup \{a_j < a_i^* \mid i \geq 2y - 2x + j\} \cup \{a_1 < a_i^*\} \tag{A.147}$$

For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.60.
It is easy to verify that the relation $>_{v,w}$ defined by Eq. A.147 is a strict total order over its domain. The relations $>_v$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ_v^* and ρ_u^*, and ρ_v^* and ρ_w^*, respectively.
From Eqs. A.53, A.60, and A.147 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho_v^*| = |\rho[v]|$ together with Eq. A.145 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

(b) *All the changes of v to b_v are done using action a_1^{b}, and then the maximally possible number of changes to w_v are done using action a_1^{w}, with the remaining changes to w_v being done using action a_2^{w}.* For this candidate for ρ_v^*, we set

$$a_i^* = \begin{cases} a_2^{\mathsf{w}} & i = 2j - 1, j \in \mathbb{N}, j \leq y - x \\ a_1^{\mathsf{w}} & i = 2j - 1, j \in \mathbb{N}, y - x < j \leq y \\ a_1^{\mathsf{b}} & \text{otherwise} \end{cases} \tag{A.148}$$

And the cost in this case is

$$x \cdot cost(a_1^{\mathsf{w}}) + (y - x) \cdot cost(a_2^{\mathsf{w}}) + y \cdot cost(a_1^{\mathsf{b}}) \tag{A.149}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho^*_w = \langle a_1, \ldots, a_{2x-1} \rangle$, we set $>_{v,w}$ according to Eq. A.137 above.
For each $u \in \mathsf{pred}(v) \setminus \{w\}$ we set $>_{v,u}$ according to Eq. A.64 above.
The relations $>_v$, $>_{v,w}$ and $>_{v,u}$ are effectively identical to the previous case. Thus, here as well, we have $>_v \cup >_{v,u}$ and $>_v \cup >_{v,w}$ forming strict partial orders over the unions of the elements of ρ^*_v and ρ^*_u, and ρ^*_v and ρ^*_w, respectively.
From Eqs. A.53, A.64, and A.137 we can now derive that any linearization of $>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w}$ defines a sequence of actions that is applicable with respect to $\{v\} \cup \mathsf{pred}(v)$. In addition, $|\rho^*_v| = |\rho[v]|$ together with Eq. A.148 implies that this action sequence provides to v the value $G[v]$ if the latter is specified.

Now, for each cost-optimal plan ρ, $\rho[v]$ cannot contain more than $y + x$ actions of both types a^{w}_1 and a^{b}_1 totally. Suppose to the contrary that $\rho[v]$ contain at least $y + x + 1$ actions of types a^{w}_1 and a^{b}_1. Then it contains no more than $y - x - 1$ actions of other types. Let $\mathsf{w}_w \cdot \mathsf{b}_w \cdot \ldots \cdot \mathsf{b}_w$ sequence of $2y$ values of w that support cost-optimal plan for v given that w can change it value any number of times. Then each action of other type will decrease the needed length of this sequence in at most 2. Therefore at most $y-x-1$ actions of other type will decrease the length in at most $2y-2x-2$, and we are left with the sequence of length $\geq 2y-(2y-2x-2) = 2x+2$. Therefore σ_w cannot support more than $y + x$ actions of types a^{w}_1 and a^{b}_1. Now, suppose that in some given cost-optimal plan $\rho[v]$ for v there are α actions of type a^{w}_1 and β actions of type a^{b}_1. Then

$$\alpha + \beta \leq y + x \tag{A.150}$$

and

$$cost(\rho_v) \geq \alpha \cdot cost(a^{\mathsf{w}}_1) + (y - \alpha) \cdot cost(a^{\mathsf{w}}_2) + \beta \cdot cost(a^{\mathsf{b}}_1) + (y - \beta) \cdot cost(a^{\mathsf{b}}_2) \tag{A.151}$$

For $(A.146) \leq (A.149)$, we have

$$cost(a^{\mathsf{b}}_2) - cost(a^{\mathsf{b}}_1) \leq cost(a^{\mathsf{w}}_2) - cost(a^{\mathsf{w}}_1) \tag{A.152}$$

Now suppose to the contrary that the plan in first case is not cost-optimal. Then from Eq. A.151 we have

$$\alpha \cdot cost(a^{\mathsf{w}}_1) + (y - \alpha) \cdot cost(a^{\mathsf{w}}_2) + \beta \cdot cost(a^{\mathsf{b}}_1) + (y - \beta) \cdot cost(a^{\mathsf{b}}_2) <$$
$$y \cdot cost(a^{\mathsf{w}}_1) + x \cdot cost(a^{\mathsf{b}}_1) + (y - x) \cdot cost(a^{\mathsf{b}}_2)$$

and from it

$$(y - \alpha) \cdot (cost(a^{\mathsf{w}}_2) - cost(a^{\mathsf{w}}_1)) < (\beta - x) \cdot (cost(a^{\mathsf{b}}_2) - cost(a^{\mathsf{b}}_1)) \tag{A.153}$$

From Eq. A.150 we have $y-\alpha \geq \beta-x$, together with Eq. A.152 contradicting Eq. A.153.
For $(A.149) \leq (A.146)$, we have

$$cost(a^{\mathsf{w}}_2) - cost(a^{\mathsf{w}}_1) \leq cost(a^{\mathsf{b}}_2) - cost(a^{\mathsf{b}}_1) \tag{A.154}$$

Now suppose to the contrary that the plan in second case is not cost-optimal. Then from Eq. A.151 we have

$$\alpha \cdot cost(a^{\mathsf{w}}_1) + (y - \alpha) \cdot cost(a^{\mathsf{w}}_2) + \beta \cdot cost(a^{\mathsf{b}}_1) + (y - \beta) \cdot cost(a^{\mathsf{b}}_2) <$$
$$x \cdot cost(a^{\mathsf{w}}_1) + (y - x) \cdot cost(a^{\mathsf{w}}_2) + y \cdot cost(a^{\mathsf{b}}_1)$$

and from it

$$(y-\beta)\cdot(cost(a_2^{\mathsf{b}})-cost(a_1^{\mathsf{b}}))<(\alpha-x)\cdot(cost(a_2^{\mathsf{w}})-cost(a_1^{\mathsf{w}})) \tag{A.155}$$

From Eq. A.150 we have $y-\beta\geq\alpha-x$, together with Eq. A.154 contradicting Eq. A.155.

(8) If $|\rho_v|=2y$, $|\sigma_w|=2x+1$, $|\sigma_w|>|\rho_v|$, then the actions of ρ_v^* are set to

$$a_i^*=\begin{cases}a_{\mathsf{w}_v|\mathsf{w}_w} & i=2j-1, j\in\mathbb{N}\\ a_{\mathsf{b}_v|\mathsf{b}_w} & \text{otherwise}\end{cases} \tag{A.156}$$

Here as well, the ordering constraints $>_v$ are set according to Eq. A.53. Likewise, if $\rho_w^*=\langle a_1,\ldots,a_{2x-1}\rangle$, we set $>_{v,w}$ according to Eq. A.110 above.
Finally, here as well, the ordering constraints $>_{v,w'}$ for the rest of the parents $w'\in\mathsf{pred}(v)\setminus\{u,w\}$ are set to empty sets.
The relations $>_v$ and $>_{v,w}$ are identical to the previous case. Thus, here as well, we have $>_v\cup>_{v,w}$ forming strict partial order over the union of the elements of ρ_v^* and ρ_w^*.

Until now, we have specified the sequences ρ_v^*, the orders $>_v$ induced by these sequences, the orders $>_{v,w}$, and proved that all $>_v\cup>_{v,w}$ and $>_w\cup>_{v,w}$ form strict partial orders over their domains. This construction allows us to apply now Theorem 22 to the (considered as sets) sequences ρ_v^* and orders $>_v$ and $>_{v,w}$, proving that

$$>=\bigcup_{v\in V}(>_v\cup\bigcup_{w\in\mathsf{pred}(v)}>_{v,w})$$

forms a strict partial order over the union of $\rho_{v_1}^*,\ldots,\rho_{v_n}^*$. Putting thing together, the above implies that any linearization ρ^* of $>$ is a plan for Π, and **post-3/2**ness of all its subsequences $\rho_{v_1}^*,\ldots,\rho_{v_n}^*$ then implies $\rho^*\in\mathcal{P}^{3/2}(\Pi)$. Moreover, if ρ is an optimal plan for Π, then $cost(\rho^*)=cost(\rho)$ implies the optimality of ρ^*. ∎

A.3.2 Construction

Given a post-3/2 action sequence ϱ from A and a variable $v\in V$, we can distinguish between the following exhaustive roles of each parent $w\in\mathsf{pred}(v)$ with respect to v along ϱ.

R1 All the actions in ϱ that change the value of v are supported by the same value of w.
That is, for some $\gamma\in\{\mathsf{b}_w,\mathsf{w}_w\}$, if $a\in\varrho_v$, then $a\in\{a_{\mathsf{b}_v|\gamma},a_{\mathsf{w}_v|\gamma}\}$.

R2 All the actions in ϱ that change the value of v to w_v are supported by the same value of w, and all the actions in ϱ that change the value of v to b_v are supported by another value of w.
That is, for some $\gamma\neq\delta\in\{\mathsf{b}_w,\mathsf{w}_w\}$, if $a\in\varrho_v$, then $a\in\{a_{\mathsf{b}_v|\gamma},a_{\mathsf{w}_v|\delta}\}$.

R3 All the actions in ϱ that change the value of v to w_v are supported by the same value of w, and none of the actions in ϱ that change the value of v to b_v are supported by w.
That is, for some $\gamma\in\{\mathsf{b}_w,\mathsf{w}_w\}$ and $\delta\notin\{\mathsf{b}_w,\mathsf{w}_w\}$, if $a\in\varrho_v$, then $a\in\{a_{\mathsf{b}_v|\delta},a_{\mathsf{w}_v|\gamma}\}$.

R4 All the actions in ϱ that change the value of v to b_v are supported by the same value of w, and none of the actions in ϱ that change the value of v to w_v are supported by w.
That is, for some $\gamma\in\{\mathsf{b}_w,\mathsf{w}_w\}$ and $\delta\notin\{\mathsf{b}_w,\mathsf{w}_w\}$, if $a\in\varrho_v$, then $a\in\{a_{\mathsf{b}_v|\gamma},a_{\mathsf{w}_v|\delta}\}$.

R5 All the actions in ϱ that change the value of v to w_v are supported by the same value of w, and all the actions in ϱ that change the value of v to b_v are supported by two values of w.
That is, for some $\gamma \neq \delta \in \{\mathsf{b}_w, \mathsf{w}_w\}$, if $a \in \varrho_v$, then $a \in \{a_{\mathsf{w}_v|\gamma}, a_{\mathsf{b}_v|\delta}, a_{\mathsf{b}_v|\gamma}\}$.

R6 All the actions in ϱ that change the value of v to b_v are supported by the same value of w, and all the actions in ϱ that change the value of v to w_v are supported by two values of w.
That is, for some $\gamma \neq \delta \in \{\mathsf{b}_w, \mathsf{w}_w\}$, if $a \in \varrho_v$, then $a \in \{a_{\mathsf{b}_v|\gamma}, a_{\mathsf{w}_v|\delta}, a_{\mathsf{w}_v|\gamma}\}$.

R7 All the actions in ϱ that change the value of v to w_v are supported by the same value of w, and some of the actions in ϱ that change the value of v to b_v are supported by another value of w and others are supported by another parent.
That is, for some $\gamma \neq \delta \in \{\mathsf{b}_w, \mathsf{w}_w\}$ and $\mu \notin \{\mathsf{b}_w, \mathsf{w}_w\}$, if $a \in \varrho_v$, then $a \in \{a_{\mathsf{w}_v|\gamma}, a_{\mathsf{b}_v|\delta}, a_{\mathsf{b}_v|\mu}\}$.

R8 All the actions in ϱ that change the value of v to b_v are supported by the same value of w, and some of the actions in ϱ that change the value of v to w_v are supported by another value of w and others are supported by another parent.
That is, for some $\gamma \neq \delta \in \{\mathsf{b}_w, \mathsf{w}_w\}$ and $\mu \notin \{\mathsf{b}_w, \mathsf{w}_w\}$, if $a \in \varrho_v$, then $a \in \{a_{\mathsf{b}_v|\gamma}, a_{\mathsf{w}_v|\delta}, a_{\mathsf{w}_v|\mu}\}$.

R9 Part of the actions in ϱ that change the value of v to b_v are supported by the same value of w, and none of the actions in ϱ that change the value of v to w_v are supported by the same value of w.

R10 Part of the actions in ϱ that change the value of v to w_v are supported by the same value of w, and none of the actions in ϱ that change the value of v to b_v are supported by the same value of w.

R11 None of the actions in ϱ are supported by w.
That is, if $a_{\alpha|\gamma} \in \varrho$, then $\gamma \notin \{\mathsf{b}_w, \mathsf{w}_w\}$.

For a given post-3/2 action sequence ϱ from A and a variable $v \in V$, each parent of v performs one of the roles R1-R11 with respect to v along ϱ, and each of the roles R1-R10 is performed by at most one of the parents of v. In addition, there are sets of roles that cannot be simultaneously performed by the parents of v with respect to v and the same action sequence ϱ, and there are roles that have to be performed in pairs. Specifically,

- If one of the roles {R1,R2,R5,R6} is played by some parent $w' \in \mathsf{pred}(v)$, then R11 must be played by all other parents $w \in \mathsf{pred}(v) \setminus \{w'\}$.
- If R3/R7/R8 is played by some parent $w_1 \in \mathsf{pred}(v)$, then R4/R9/R10, respectively, must be played by some parent $w_2 \in \mathsf{pred}(v) \setminus \{w_1\}$, and R11 must be played by all other parents $w \in \mathsf{pred}(v) \setminus \{w_1, w_2\}$.

Considering a variable v and its parents $\mathsf{pred}(v)$ through the lens of these eleven roles, suppose we now aim at assigning these roles to $\mathsf{pred}(v)$ by considering them one after another in some arbitrary order. Given the aforementioned constraints on the role assignment, at each step of this sequential process we can be in one of the following eight states, with the whole process being described by a state machine depicted in Fig. A.6.

S1 All the roles R1-R11 are still available (to be assigned to the parents of v).

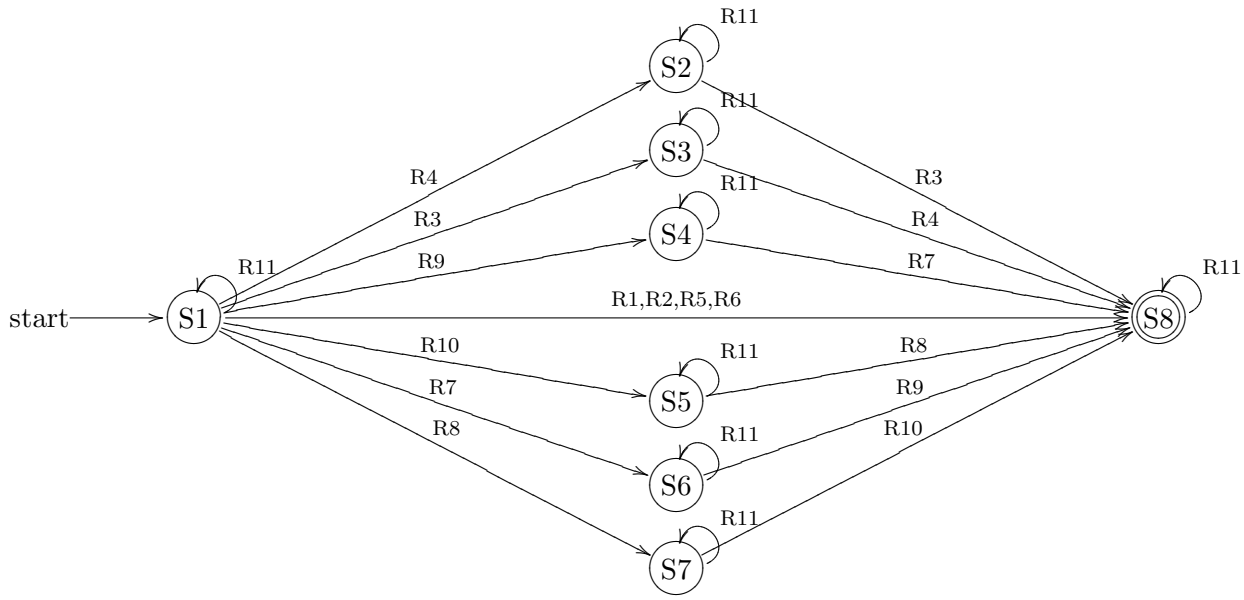


Figure A.6: State machine describing the process of "sequential" role assignment to the parents of v (with respect to v). Each transition is labeled with a set of roles, one of whose is getting assigned to a parent of v at the corresponding step.

S2 Only roles {R3,R11} are still available.

S3 Only roles {R4,R11} are available.

S4 Only roles {R7,R11} are available.

S5 Only roles {R8,R11} are available.

S6 Only roles {R9,R11} are available.

S7 Only roles {R10,R11} are available.

S8 Only role R11 is available.

Given this language of "roles" and "states", we now proceed with specifying our constraint optimization problem $\mathsf{COP}_\Pi = (\mathcal{X}, \mathcal{F})$ for a task $\Pi = \langle V, A, I, G, \mathit{cost} \rangle \in \mathbf{P}(1)$. In what follows, for each variable $v \in V$, we assume a fixed (*arbitrary* chosen) numbering $\{w_1, \ldots, w_k\}$ of $\mathsf{pred}(v)$ with respect to v.

1. Similarly to the uniform-cost case, the variable set $\mathcal{X}$ contains a variable x_v for each planning variable $v \in V$, and a variable x_v^w for each edge $(w, v) \in CG(\Pi)$. That is,

$$\begin{aligned} \mathcal{X} &= \mathcal{X}^V \cup \mathcal{X}^{\mathcal{E}} \\ \mathcal{X}^V &= \{x_v \mid v \in V\} \\ \mathcal{X}^{\mathcal{E}} &= \{x_v^w \mid (w, v) \in CG(\Pi)\} \end{aligned} \tag{A.157}$$

2. For each variable $x_v \in \mathcal{X}^V$, the domain $\mathcal{D}(x_v)$ consists of all possible valid prefixes of $\sigma(v)$. For each variable $x_v^{w_i} \in \mathcal{X}^{\mathcal{E}}$, the domain $\mathcal{D}(x_v^{w_i})$ consists of all possible quadruples satisfying Eq. A.158.

$$\begin{aligned} \mathcal{D}(x_v) &= \{\sigma_v \in \trianglerighteq^*[\sigma(v)]\} \\ \mathcal{D}(x_v^{w_i}) &= \left\{ [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!] \;\middle|\; \begin{array}{l} 0 \le \eta \le n, 0 \le \#_{\mathsf{w}}, \#_{\mathsf{b}} \le \lceil \frac{\eta}{2} \rceil \\ \mathbf{S} \in \{\mathrm{S}_1, \ldots, \mathrm{S}_8\} \end{array} \right\} \end{aligned} \tag{A.158}$$

The semantics of Eq. A.158 is as follows. Let $\{w_1, \ldots, w_k\}$ be an arbitrary fixed ordering of $\mathsf{pred}(v)$. If x_v takes the value $\sigma_v \in \mathcal{D}(x_v)$, then v is forced to provide the sequence of values σ_v. In turn, if $x_v^{w_i}$ takes the value $[\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!]$, then η corresponds to the number of value changes of v, $\#_{\mathsf{w}}$ and $\#_{\mathsf{b}}$ correspond to the number of value changes of v to w_v and b_v, respectively, that should be performed by the actions prevailed by the values of $\{w_1, \ldots, w_i\}$, and the state-component $\mathbf{S}$ captures the roles that can be assigned to the parents $\{w_1, \ldots, w_i\}$.

3. Similarly to the uniform-cost case, for each variable $x \in \mathcal{X}$, the set $\mathcal{F}$ contains a non-negative, real-valued function φ_x with the scope

$$Q_x = \begin{cases} \{x_v\}, & x = x_v, k = 0 \\ \{x_v, x_v^{w_k}\}, & x = x_v, k > 0 \\ \{x_v^{w_1}, x_{w_1}\}, & x = x_v^{w_1}, k > 0 \\ \{x_v^{w_j}, x_v^{w_{j-1}}, x_{w_j}\}, & x = x_v^{w_j}, 1 < j \le k \end{cases} \tag{A.159}$$

where $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$ (with $k = 0$ meaning $\mathsf{pred}(v) = \emptyset$).

Proceeding now with specifying the functional components $\mathcal{F}$ of COP_Π, first, for each x_v with $\mathsf{pred}(v) = \emptyset$, and for each $\sigma_v \in \trianglerighteq^*[v]$, we set $\varphi_{x_v}(\sigma_v)$ according to Eq. A.160.

$$\varphi_{x_v}(\sigma_v) = \begin{cases} 0, & |\sigma_v| = 1, \\ cost(a_{\mathsf{w}_v}), & |\sigma_v| = 2, a_{\mathsf{w}_v} \in A_v, \\ \lceil \frac{|\sigma_v|-1}{2} \rceil \cdot cost(a_{\mathsf{w}_v}) + \lfloor \frac{|\sigma_v|-1}{2} \rfloor \cdot cost(a_{\mathsf{b}_v}), & |\sigma_v| > 2, a_{\mathsf{w}_v}, a_{\mathsf{b}_v} \in A_v, \\ \infty, & \text{otherwise} \end{cases} \tag{A.160}$$

In turn, for each planning variable $v \in V$ with $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$, $k > 0$, the function φ_{x_v} is set as in Eq. A.161.

$$\varphi_{x_v}(\sigma_v, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!]) = \begin{cases} 0, & |\sigma_v| = 1, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!] = [\![\mathrm{S8}, 0, 0, 0]\!], \\ 0, & |\sigma_v| > 1, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!] = [\![\mathrm{S1}, \lceil \frac{|\sigma_v|-1}{2} \rceil, \lfloor \frac{|\sigma_v|-1}{2} \rfloor, |\sigma_v| - 1]\!], \\ \infty, & \text{otherwise} \end{cases} \tag{A.161}$$

The semantics of Eq. A.161 is simple—if no value changes of v are required, then trivially no support of $\mathsf{pred}(v)$ to v is needed; otherwise, all possible roles for $\mathsf{pred}(v)$ should be considered.

Now, we proceed with specifying a *generic* function φ that, for each $v \in V$, each $w \in \mathsf{pred}(v)$, and each $(\mathrm{R}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) \in \{\mathrm{R1}, \ldots, \mathrm{R10}\} \times \mathcal{D}(x_v^w) \times \mathcal{D}(x_w)$, provides the marginal over

the actions A_v cost of w taking the role R, and under this role, supporting $\#_{\mathsf{w}}$ changes of v to w_v and $\#_{\mathsf{b}}$ changes of v to b_v, out of total η changes of v needed. For ease of presentation, let $\xi(x_1, x_2, y_1, y_2)$ denote the cost of an action sequence consisting of x_1 actions of type $a_{\mathsf{w}_v|\mathsf{b}_w}$, x_2 actions of type $a_{\mathsf{w}_v|\mathsf{w}_w}$, y_1 actions of type $a_{\mathsf{b}_v|\mathsf{w}_w}$, y_2 actions of type $a_{\mathsf{b}_v|\mathsf{b}_w}$, that is

$$\xi(x_1, x_2, y_1, y_2) = x_1 \cdot cost(a_{\mathsf{w}_v|\mathsf{b}_w}) + x_2 \cdot cost(a_{\mathsf{w}_v|\mathsf{w}_w}) + y_1 \cdot cost(a_{\mathsf{b}_v|\mathsf{w}_w}) + y_2 \cdot cost(a_{\mathsf{b}_v|\mathsf{b}_w}) \quad \text{(A.162)}$$

While the notation $\xi_{v,w}$ is probably more appropriate for the semantics of ξ, we adopt the latter for its shortness because the identity of v and w will always be clear from the context.

The Eqs. A.163-A.172 below specify $\varphi\,(\mathrm{R}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w)$ for $\mathrm{R} \in \{\mathrm{R1}, \ldots, \mathrm{R10}\}$. The semantics of $\varphi\,(\mathrm{R}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w)$ is to capture the minimal accumulative cost over the actions from A_v to achieve $\#_{\mathsf{w}}$ and $\#_{\mathsf{b}}$ (out of η) value changes of v under the support of the parent w playing the role R with respect to v. For example, role R3 means supporting all the actions that change the value of v to w_v, and Eq. A.165 gives us the minimal cost of this support in terms of the accumulative cost of the supported actions from A_v. These minimal costs are taken from the relevant cases in the proof of Theorem 26, notably

(Eq. A.163) Case (I).

(Eq. A.164) Cases $\{(III), (IV)\}.\{2, 4, 6, 8\}$.

(Eq. A.165) Case (II), the cost of all actions that change the value of v to w_v.

(Eq. A.166) Case (II), the cost of all actions that change the value of v to b_v.

(Eq. A.167) Cases $\{(III), (IV)\}.\{1, 3, 5, 7\}.a$, the minimal cost.

(Eq. A.168) Cases $\{(III), (IV)\}.\{1, 3, 5, 7\}.b$, the minimal cost.

(Eq. A.169) Cases $\{(III), (IV)\}.\{1, 3, 5, 7\}.a$, the cost of all actions prevailed by one parent.

(Eq. A.170) Cases $\{(III), (IV)\}.\{1, 3, 5, 7\}.b$, the cost of all actions prevailed by one parent.

(Eq. A.171) The residue of cases $\{(III), (IV)\}.\{1, 3, 5, 7\}.a$. (Together with Eq. A.169 this gives us the full cost of changing v as required.)

(Eq. A.172) The residue of cases $\{(III), (IV)\}.\{1, 3, 5, 7\}.b$. (Together with Eq. A.169 this gives us the full cost of changing v as required.)

$$\varphi\,(\mathrm{R1}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) = \begin{cases} \xi(\#_{\mathsf{w}}, 0, 0, \#_{\mathsf{b}}), & |\sigma_w| = 1, \#_{\mathsf{w}} = \lceil \frac{\eta}{2} \rceil, \#_{\mathsf{b}} = \lfloor \frac{\eta}{2} \rfloor \\ \min \left\{ \begin{array}{l} \xi(\#_{\mathsf{w}}, 0, 0, \#_{\mathsf{b}}), \\ \xi(0, \#_{\mathsf{w}}, \#_{\mathsf{b}}, 0) \end{array} \right\}, & |\sigma_w| > 1, \#_{\mathsf{w}} = \lceil \frac{\eta}{2} \rceil, \#_{\mathsf{b}} = \lfloor \frac{\eta}{2} \rfloor \\ \infty, & \text{otherwise} \end{cases} \quad \text{(A.163)}$$

$$\varphi(\mathrm{R2}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) = \begin{cases} \xi(\#_{\mathsf{w}}, 0, \#_{\mathsf{b}}, 0), & |\sigma_w| = \eta \geq 2, \#_{\mathsf{w}} = \lceil \frac{\eta}{2} \rceil, \#_{\mathsf{b}} = \lfloor \frac{\eta}{2} \rfloor \\ \min \left\{ \begin{array}{l} \xi(\#_{\mathsf{w}}, 0, \#_{\mathsf{b}}, 0), \\ \xi(0, \#_{\mathsf{w}}, 0, \#_{\mathsf{b}}) \end{array} \right\}, & |\sigma_w| > \eta \geq 2, \#_{\mathsf{w}} = \lceil \frac{\eta}{2} \rceil, \#_{\mathsf{b}} = \lfloor \frac{\eta}{2} \rfloor \\ \infty, & \text{otherwise} \end{cases} \quad \text{(A.164)}$$

$$\varphi(\mathrm{R3}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) = \begin{cases} \#_{\mathsf{w}} \cdot cost(a_{\mathsf{w}_v|\mathsf{b}_w}), & |\sigma_w| = 1, \#_{\mathsf{w}} = \lceil \frac{\eta}{2} \rceil, \#_{\mathsf{b}} = 0 \\ \min \left\{ \begin{array}{l} \#_{\mathsf{w}} \cdot cost(a_{\mathsf{w}_v|\mathsf{b}_w}), \\ \#_{\mathsf{w}} \cdot cost(a_{\mathsf{w}_v|\mathsf{w}_w}) \end{array} \right\}, & |\sigma_w| > 1, \#_{\mathsf{w}} = \lceil \frac{\eta}{2} \rceil, \#_{\mathsf{b}} = 0 \\ \infty, & \text{otherwise} \end{cases} \tag{A.165}$$

$$\varphi(\mathrm{R4}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) = \begin{cases} \#_{\mathsf{b}} \cdot cost(a_{\mathsf{b}_v|\mathsf{b}_w}), & |\sigma_w| = 1, \#_{\mathsf{w}} = 0, \#_{\mathsf{b}} = \lfloor \frac{\eta}{2} \rfloor \\ \min \left\{ \begin{array}{l} \#_{\mathsf{b}} \cdot cost(a_{\mathsf{b}_v|\mathsf{b}_w}), \\ \#_{\mathsf{b}} \cdot cost(a_{\mathsf{b}_v|\mathsf{w}_w}) \end{array} \right\}, & |\sigma_w| > 1, \#_{\mathsf{w}} = 0, \#_{\mathsf{b}} = \lfloor \frac{\eta}{2} \rfloor \\ \infty, & \text{otherwise} \end{cases} \tag{A.166}$$

$$\varphi(\mathrm{R5}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) = \begin{cases} \min \left\{ \begin{array}{l} \xi(y+1, 0, x-1, y-x+1), \\ \xi(0, y+1, y-x+1, x-1) \end{array} \right\}, & \eta = 2y+1, |\sigma_w| = 2x, 1 < x \leq y, \\ & \#_{\mathsf{w}} = y+1, \#_{\mathsf{b}} = y \\ \min \left\{ \begin{array}{l} \xi(y, 0, x, y-x), \\ \xi(0, y, y-x+1, x-1) \end{array} \right\}, & \eta = 2y, |\sigma_w| = 2x, 1 < x < y, \\ & \#_{\mathsf{w}} = \#_{\mathsf{b}} = y \\ \xi(y, 0, 1, y-1), & \eta = 2y, |\sigma_w| = 2, 1 < y, \\ & \#_{\mathsf{w}} = \#_{\mathsf{b}} = y \\ \xi(0, y, 1, y-1), & \eta = |\sigma_w| = 2y, 1 < y, \\ & \#_{\mathsf{w}} = \#_{\mathsf{b}} = y \\ \min \left\{ \begin{array}{l} \xi(y+1, 0, x, y-x), \\ \xi(0, y+1, y-x+1, x-1) \end{array} \right\}, & \eta = 2y+1, |\sigma_w| = 2x+1, 1 < x < y, \\ & \#_{\mathsf{w}} = y+1, \#_{\mathsf{b}} = y \\ \xi(y+1, 0, 1, y-1), & \eta = 2y+1, |\sigma_w| = 3, 1 < y, \\ & \#_{\mathsf{w}} = y+1, \#_{\mathsf{b}} = y \\ \xi(0, y+1, 1, y-1), & \eta = |\sigma_w| = 2y+1, 1 < y, \\ & \#_{\mathsf{w}} = y+1, \#_{\mathsf{b}} = y \\ \min \left\{ \begin{array}{l} \xi(y, 0, x, y-x), \\ \xi(0, y, y-x, x) \end{array} \right\}, & \eta = 2y, |\sigma_w| = 2x+1, 1 \leq x < y, \\ & \#_{\mathsf{w}} = \#_{\mathsf{b}} = y \\ \infty, & \text{otherwise} \end{cases} \tag{A.167}$$

$$\varphi(\mathrm{R6}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) = \begin{cases} \min\left\{ \begin{array}{l} \xi(x, y+1-x, y, 0), \\ \xi(y+1-x, x, 0, y) \end{array} \right\}, & \eta = 2y+1, |\sigma_w| = 2x, 1 \le x \le y, \\ & \#_{\mathsf{w}} = y+1, \#_{\mathsf{b}} = y \\ \min\left\{ \begin{array}{l} \xi(x, y-x, y, 0), \\ \xi(y-x+1, x-1, 0, y) \end{array} \right\}, & \eta = 2y, |\sigma_w| = 2x, 1 < x < y, \\ & \#_{\mathsf{w}} = \#_{\mathsf{b}} = y \\ \xi(1, y-1, y, 0), & \eta = 2y, |\sigma_w| = 2, 1 < y, \\ & \#_{\mathsf{w}} = \#_{\mathsf{b}} = y \\ \xi(1, y-1, 0, y), & \eta = |\sigma_w| = 2y, 1 < y, \\ & \#_{\mathsf{w}} = \#_{\mathsf{b}} = y \\ \min\left\{ \begin{array}{l} \xi(x+1, y-x, y, 0), \\ \xi(y-x+1, x, 0, y) \end{array} \right\}, & \eta = 2y+1, |\sigma_w| = 2x+1, 1 \le x < y, \\ & \#_{\mathsf{w}} = y+1, \#_{\mathsf{b}} = y \\ \xi(1, y, 0, y), & \eta = |\sigma_w| = 2y+1, 1 \le y, \\ & \#_{\mathsf{w}} = y+1, \#_{\mathsf{b}} = y \\ \min\left\{ \begin{array}{l} \xi(x, y-x, y, 0), \\ \xi(y-x, x, 0, y) \end{array} \right\}, & \eta = 2y, |\sigma_w| = 2x+1, 1 \le x < y, \\ & \#_{\mathsf{w}} = \#_{\mathsf{b}} = y \\ \infty, & \text{otherwise} \end{cases} \tag{A.168}$$

$$\varphi(\mathrm{R7}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) = \begin{cases} \min\left\{ \begin{array}{l} \xi(y+1, 0, x-1, 0), \\ \xi(0, y+1, 0, x-1) \end{array} \right\}, & \eta = 2y+1, |\sigma_w| = 2x, 1 < x \le y, \\ & \#_{\mathsf{w}} = y+1, \#_{\mathsf{b}} = x-1 \\ \xi(y, 0, x, 0), & cost(a_{\mathsf{w}_v|\mathsf{b}_w}) < cost(a_{\mathsf{w}_v|\mathsf{w}_w}), \\ & \eta = 2y, |\sigma_w| = 2x, 1 \le x < y, \\ & \#_{\mathsf{w}} = y, \#_{\mathsf{b}} = x \\ \xi(0, y, 0, x-1), & cost(a_{\mathsf{w}_v|\mathsf{b}_w}) \ge cost(a_{\mathsf{w}_v|\mathsf{w}_w}), \\ & \eta = 2y, |\sigma_w| = 2x, 1 < x \le y, \\ & \#_{\mathsf{w}} = y, \#_{\mathsf{b}} = x-1 \\ \xi(y+1, 0, x, 0), & cost(a_{\mathsf{w}_v|\mathsf{b}_w}) < cost(a_{\mathsf{w}_v|\mathsf{w}_w}), \\ & \eta = 2y+1, |\sigma_w| = 2x+1, 1 \le x < y, \\ & \#_{\mathsf{w}} = y+1, \#_{\mathsf{b}} = x \\ \xi(0, y+1, 0, x-1), & cost(a_{\mathsf{w}_v|\mathsf{b}_w}) \ge cost(a_{\mathsf{w}_v|\mathsf{w}_w}), \\ & \eta = 2y+1, |\sigma_w| = 2x+1, 1 < x \le y, \\ & \#_{\mathsf{w}} = y+1, \#_{\mathsf{b}} = x-1 \\ \min\left\{ \begin{array}{l} \xi(y, 0, x, 0), \\ \xi(0, y, 0, x) \end{array} \right\}, & \eta = 2y, |\sigma_w| = 2x+1, 1 \le x < y, \\ & \#_{\mathsf{w}} = y, \#_{\mathsf{b}} = x \\ \infty, & \text{otherwise} \end{cases} \tag{A.169}$$

$$\varphi(\mathrm{R8}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) = \begin{cases} \min\left\{ \begin{array}{l} \xi(x,0,y,0), \\ \xi(0,x,0,y) \end{array} \right\}, & \eta = 2y+1, |\sigma_w| = 2x, 1 \le x \le y, \\ & \#_{\mathsf{w}} = x, \#_{\mathsf{b}} = y \\ \xi(x,0,y,0), & cost(a_{\mathsf{w}_v|\mathsf{b}_w}) < cost(a_{\mathsf{w}_v|\mathsf{w}_w}), \\ & \eta = 2y, |\sigma_w| = 2x, 1 \le x < y, \\ & \#_{\mathsf{w}} = x, \#_{\mathsf{b}} = y \\ \xi(0,x-1,0,y), & cost(a_{\mathsf{w}_v|\mathsf{b}_w}) \ge cost(a_{\mathsf{w}_v|\mathsf{w}_w}), \\ & \eta = 2y, |\sigma_w| = 2x, 1 < x \le y, \\ & \#_{\mathsf{w}} = x-1, \#_{\mathsf{b}} = y \\ \xi(x+1,0,y,0), & \eta = 2y+1, |\sigma_w| = 2x+1, 1 \le x < y, \\ & \#_{\mathsf{w}} = x+1, \#_{\mathsf{b}} = y \\ \xi(0,x,0,y), & \eta = 2y+1, |\sigma_w| = 2x+1, 1 \le x \le y, \\ & \#_{\mathsf{w}} = x, \#_{\mathsf{b}} = y \\ \min\left\{ \begin{array}{l} \xi(x,0,y,0), \\ \xi(0,x,0,y) \end{array} \right\}, & \eta = 2y, |\sigma_w| = 2x+1, 1 \le x < y, \\ & \#_{\mathsf{w}} = x, \#_{\mathsf{b}} = y \\ \infty, & \text{otherwise} \end{cases} \tag{A.170}$$

$$\varphi(\mathrm{R9}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) = \begin{cases} \#_{\mathsf{b}} \cdot cost(a_{\mathsf{b}_v|\mathsf{w}_w}), & |\sigma_w| = 1, \#_{\mathsf{w}} = 0, \#_{\mathsf{b}} < \lfloor \frac{\eta}{2} \rfloor \\ \min\left\{ \begin{array}{l} \#_{\mathsf{b}} \cdot cost(a_{\mathsf{b}_v|\mathsf{w}_w}), \\ \#_{\mathsf{b}} \cdot cost(a_{\mathsf{b}_v|\mathsf{b}_w}) \end{array} \right\}, & |\sigma_w| > 1, \#_{\mathsf{w}} = 0, \#_{\mathsf{b}} < \lfloor \frac{\eta}{2} \rfloor \\ \infty, & \text{otherwise} \end{cases} \tag{A.171}$$

$$\varphi(\mathrm{R10}, [\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], \sigma_w) = \begin{cases} \#_{\mathsf{w}} \cdot cost(a_{\mathsf{w}_v|\mathsf{b}_w}), & |\sigma_w| = 1, \#_{\mathsf{w}} < \lceil \frac{\eta}{2} \rceil, \#_{\mathsf{b}} = 0 \\ \min\left\{ \begin{array}{l} \#_{\mathsf{w}} \cdot cost(a_{\mathsf{w}_v|\mathsf{b}_w}), \\ \#_{\mathsf{w}} \cdot cost(a_{\mathsf{w}_v|\mathsf{w}_w}) \end{array} \right\}, & |\sigma_w| > 1, \#_{\mathsf{w}} < \lceil \frac{\eta}{2} \rceil, \#_{\mathsf{b}} = 0 \\ \infty, & \text{otherwise} \end{cases} \tag{A.172}$$

Having specified the function φ, we now use it, in particular, for specifying the functional component $\varphi_{x_v^{w_1}}$ as in Eq. A.173. This equation actually *emulates movements in the state machine for v as in Figure A.6 to the terminal state S*8.

$$\varphi_{x_v^{w_1}}([\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1}) = \begin{cases} \min\left\{\begin{array}{l}\varphi(\mathrm{R1},[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1}),\\ \varphi(\mathrm{R2},[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1}),\\ \varphi(\mathrm{R5},[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1}),\\ \varphi(\mathrm{R6},[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1})\end{array}\right\}, & \mathbf{S}=\mathrm{S1},\\ \varphi(\mathrm{R3},[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1}), & \mathbf{S}=\mathrm{S2},\\ \varphi(\mathrm{R4},[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1}), & \mathbf{S}=\mathrm{S3},\\ \varphi(\mathrm{R7},[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1}), & \mathbf{S}=\mathrm{S4},\\ \varphi(\mathrm{R8},[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1}), & \mathbf{S}=\mathrm{S5},\\ \varphi(\mathrm{R9},[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1}), & \mathbf{S}=\mathrm{S6},\\ \varphi(\mathrm{R10},[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!],\sigma_{w_1}), & \mathbf{S}=\mathrm{S7},\\ 0, & \begin{array}{l}\mathbf{S}=\mathrm{S8},\\ \#_{\mathsf{w}}=0,\\ \#_{\mathsf{b}}=0\end{array}\\ \infty, & \text{otherwise}\end{cases} \tag{A.173}$$

We now proceed with the rest of the functional components $\varphi_{x_v^{w_2}},\ldots,\varphi_{x_v^{w_k}}$. For each $2\le j\le k$, each $[\![\mathbf{S},\#_{\mathsf{w}},\#_{\mathsf{b}},\eta]\!]\in\mathcal{D}(x_v^{w_j})$, each $[\![\mathbf{S}',\#'_{\mathsf{w}},\#'_{\mathsf{b}},\eta']\!]\in\mathcal{D}(x_v^{w_{j-1}})$, and each $\sigma_w\in\mathcal{D}(x_{w_j})=\trianglerighteq^*[w_j]$, the value of $\varphi_{x_v^{w_j}}$ is set according to Eq. A.174. *This equation also emulates movements in the state machine for v as in* Figure A.6*—each sub-case of* Eq. A.174 *deals with a certain transition in that state machine.*

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) =$$

$$\begin{cases}
\min \left\{ \begin{array}{l} \varphi(\mathrm{R1}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), \\ \varphi(\mathrm{R2}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), \\ \varphi(\mathrm{R5}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), \\ \varphi(\mathrm{R6}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}) \end{array} \right\}, & \mathbf{S} = \mathrm{S1}, \mathbf{S}' = \mathrm{S8}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R4}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S1}, \mathbf{S}' = \mathrm{S2}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R3}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S1}, \mathbf{S}' = \mathrm{S3}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R9}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S1}, \mathbf{S}' = \mathrm{S4}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R10}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S1}, \mathbf{S}' = \mathrm{S5}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R7}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S1}, \mathbf{S}' = \mathrm{S6}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R8}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S1}, \mathbf{S}' = \mathrm{S7}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R3}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S2}, \mathbf{S}' = \mathrm{S8}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R4}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S3}, \mathbf{S}' = \mathrm{S8}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R7}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S4}, \mathbf{S}' = \mathrm{S8}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R8}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S5}, \mathbf{S}' = \mathrm{S8}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R9}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S6}, \mathbf{S}' = \mathrm{S8}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
\varphi(\mathrm{R10}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}), & \mathbf{S} = \mathrm{S7}, \mathbf{S}' = \mathrm{S8}, \eta = \eta', \\ & \#_{\mathsf{w}} \geq \#'_{\mathsf{w}}, \#_{\mathsf{b}} \geq \#'_{\mathsf{b}} \\
0, & \mathbf{S} = \mathbf{S}', \eta = \eta', \\ & \#_{\mathsf{w}} = \#'_{\mathsf{w}}, \#_{\mathsf{b}} = \#'_{\mathsf{b}} \\
\infty, & \text{otherwise}
\end{cases} \tag{A.174}$$

This finalizes the construction of COP_Π, and this construction constitutes the first three steps of the algorithm polytree-1-dep in Figure A.7(a). The subsequent steps of this algorithm are conceptually similar to these of the polytree-1-dep-uniform algorithm in Section A.2. It is not hard to verify from Eqs. A.157-A.159, and the fact that the causal graph of $\Pi \in \mathbf{P}(1)$ forms a polytree that

(i) for each variable $x \in \mathcal{X}$, $|\mathcal{D}(x)| = poly(n)$,

(ii) the tree-width of the cost network of $\mathcal{F}$ is ≤ 3, and

(iii) the optimal tree-decomposition of the COP_Π's cost network is given by any topological ordering of the causal graph that is consistent with the (arbitrary yet fixed at the time of the COP_Π's construction) orderings of each planning variable's parents in the causal graph.

procedure polytree-1-dep($\Pi = \langle V, A, I, G, cost \rangle$)
takes a planning task $\Pi \in \mathbf{P}(1)$
returns a cost-optimal plan for Π if Π is solvable, and fails otherwise
create a set of variables $\mathcal{X}$ as in Eqs. A.157-A.158
create a set of functions $\mathcal{F} = \{\varphi_x \mid x \in \mathcal{X}\}$ with scopes as in Eq. A.159
for each $x \in \mathcal{X}$ **do**
specify φ_x according to Eqs. A.160-A.174
endfor
set $\mathsf{COP}_\Pi := (\mathcal{X}, \mathcal{F})$ with global objective min $\sum_{\varphi \in \mathcal{F}} \varphi(\mathcal{X})$
$\overline{x} :=$ solve-tree-cop(COP_Π)
if $\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}) = \infty$ **then return** failure
extract plan ρ from $\overline{x}$ with $cost(\rho) = \sum_{\varphi \in \mathcal{F}} \varphi(\overline{x})$
return ρ

Figure A.7: Algorithm for cost-optimal planning for $\mathbf{P}(1)$ tasks.

A.3.3 Correctness and Complexity

Lemma 11 *Let Π be a $\mathbf{P}(1)$ task, $\mathsf{COP}_\Pi = (\mathcal{X}, \mathcal{F})$ be the corresponding constraint optimization problem, and $\overline{x}$ be an optimal assignment to $\mathcal{X}$ with $\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}) = \alpha$.*

(I) If $\alpha < \infty$, then a plan of cost α for Π can be reconstructed from $\overline{x}$ in time polynomial in the description size of Π.

(II) If Π has a plan, then $\alpha < \infty$.

Proof:
(I) Given a COP solution $\overline{x}$ with $\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}) = \alpha < \infty$, we construct a plan ρ for Π with $cost(\rho) = \alpha$. We construct this plan by

1. Traversing the planning variables in a topological ordering of the causal graph $CG(\Pi)$, and associating each variable v with a sequence $\rho_v \in A_v^*$.

2. Merging the constructed sequences $\rho_{v_1}, \ldots, \rho_{v_n}$ into the desired plan ρ.

For each $v \in V$ with $\mathsf{pred}(v) = \emptyset$ we set $\rho_v = \langle a_1 \cdot \ldots \cdot a_l \rangle$, where $l = |\overline{x}_v| - 1$, and a_i is defined as in Eq A.175 below.

$$a_i = \begin{cases} a_{\mathsf{w}_v}, & i \text{ is odd}, \\ a_{\mathsf{b}_v}, & i \text{ is even}, \end{cases} \tag{A.175}$$

Note that Eq. A.175 is well-defined—the existence of the essential for Eq. A.175 actions $a_{\mathsf{w}_v}/a_{\mathsf{b}_v}$ is implied by Eq. A.160 and $\alpha < \infty$.

In turn, for each $v \in V$ with $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$, given $\overline{x}_v^{w_1}, \ldots, \overline{x}_v^{w_k}$, we distinguish between the following cases.

[**R**1 **is played**] R1 is played by one of the parents, while all other parents play role R11.

[**R**1 **is played by** w_1] Eq. A.173 then implies

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = \varphi(\mathrm{R1}, \overline{x}_v^{w_1}, \overline{x}_{w_1})$$

and $\overline{x}_v^{w_1} \ni \mathbf{S} = \mathrm{S1}$. From Eq. A.174 we then have $\overline{x}_v^{w_j} \ni \mathbf{S}' = \mathrm{S1}$ for each $1 < j \leq k$, giving us

$$\varphi_{x_v^{w_j}}(\overline{x}_v^{w_j}, \overline{x}_v^{w_{j-1}}, \overline{x}_{w_j}) = 0.$$

[**R**1 **is played by** w_j**,** $j > 1$] Eq. A.174 then implies

$$\varphi_{x_v^{w_j}}(\llbracket \mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta \rrbracket, \llbracket \mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta' \rrbracket, \sigma_{w_j}) = \varphi(\mathrm{R1}, \llbracket \mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta \rrbracket, \sigma_{w_j})$$

and, for all $1 < i \neq j \leq k$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

From Eq. A.173 we also have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = 0.$$

In both these sub-cases, ρ_v, $>_v$ and $>_{v,w}$ are specified as in the proof of Theorem 26, case I.

[**R**2 **is played**] R2 is played by one of the parents, while all other parents play role R11.

[**R**2 **is played by** w_1] Eq. A.173 then implies

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = \varphi(\mathrm{R2}, \overline{x}_v^{w_1}, \overline{x}_{w_1})$$

and, for each $1 < j \leq k$, Eq. A.174 implies

$$\varphi_{x_v^{w_j}}(\overline{x}_v^{w_j}, \overline{x}_v^{w_{j-1}}, \overline{x}_{w_j}) = 0.$$

If $\varphi(\mathrm{R2}, \llbracket \mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta \rrbracket, \sigma_{w_1}) = \xi(\#_{\mathsf{w}}, 0, \#_{\mathsf{b}}, 0)$, then ρ_v, $>_v$ and $>_{v,w}$ are specified as in the proof of Theorem 26, case III.2, otherwise, as in the case IV.2.

[**R**2 **is played by** w_j**,** $j > 1$] Eq. A.174 then implies

$$\varphi_{x_v^{w_j}}(\llbracket \mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta \rrbracket, \llbracket \mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta' \rrbracket, \sigma_{w_j}) = \varphi(\mathrm{R2}, \llbracket \mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta \rrbracket, \sigma_{w_j})$$

and, for all $1 < i \neq j \leq k$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

From Eq. A.173 we also have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = 0.$$

If $\varphi(\mathrm{R2}, \llbracket \mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta \rrbracket, \sigma_{w_j}) = \xi(\#_{\mathsf{w}} - \#'_{\mathsf{w}}, 0, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, 0)$, then ρ_v, $>_v$ and $>_{v,w}$ are specified as in the proof of Theorem 26, case III.2, otherwise, as in the case IV.2.

[**R**3 **and R**4 **are played**] Those roles are played by two of the parents, while all other parents play role R11.

[**R3 is played by w_1, R4 is played by w_j, $j > 1$**] From Eqs. A.173 and A.174 we then have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = \varphi(\mathrm{R3}, \overline{x}_v^{w_1}, \overline{x}_{w_1}),$$

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\mathrm{R4}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

and, for all $1 < i \leq k$, such that $i \neq j$:

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

[**R4 is played by w_1, R3 is played by w_j, $j > 1$**] From Eqs. A.173 and A.174 we then have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = \varphi(\mathrm{R4}, \overline{x}_v^{w_1}, \overline{x}_{w_1}),$$

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\mathrm{R3}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

and, for all $1 < i \neq j \leq k$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

[**R3 is played by w_j, R4 is played by w_t, $j \neq t$, $j, t > 1$**] From Eqs. A.173 and A.174 we have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = 0,$$

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\mathrm{R3}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

$$\varphi_{x_v^{w_t}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_t}) = \varphi(\mathrm{R4}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_t}),$$

and, for all $1 < i \leq k$ such that $i \notin \{j, t\}$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0$$

In all these three sub-cases, ρ_v, $>_v$ and $>_{v,w}$ are specified as in the proof of Theorem 26, case II.

[**R5 is played**] R5 is played by one of the parents, while all other parents play role R11.

[**R5 is played by w_1**] Eqs. A.173 and A.174 imply

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = \varphi(\mathrm{R5}, \overline{x}_v^{w_1}, \overline{x}_{w_1}),$$

and, for each $1 < j \leq k$,

$$\varphi_{x_v^{w_j}}(\overline{x}_v^{w_j}, \overline{x}_v^{w_{j-1}}, \overline{x}_{w_j}) = 0.$$

Considering now the specification of the function φ in Eq. A.167,

- If the first case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.1.a.
- If the first case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.1.a.
- If the second case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.3.a.

- If the second case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.3.a.
- If the third case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.3.a.
- If the forth case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.3.a.
- If the fifth case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.5.a.
- If the fifth case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.5.a.
- If the sixth case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.5.a.
- If the seventh case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.5.a.
- If the eighth case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.7.a.
- If the eighth case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.7.a.

[**R5 is played by** w_j**,** $j > 1$] Eq. A.174 then implies

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\mathrm{R5}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

and, for all $1 < i \neq j \leq k$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

From Eq. A.173 we also have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = 0.$$

Here ρ_v, $>_v$ and $>_{v,w}$ are specified exactly as in the previous case.

[**R6 is played**] R6 is played by one of the parents, while all other parents play role R11.

[**R6 is played by** w_1] Eqs. A.173 and A.174 imply

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = \varphi(\mathrm{R6}, \overline{x}_v^{w_1}, \overline{x}_{w_1}),$$

and, for each $1 < j \leq k$,

$$\varphi_{x_v^{w_j}}(\overline{x}_v^{w_j}, \overline{x}_v^{w_{j-1}}, \overline{x}_{w_j}) = 0.$$

Considering now the specification of the function φ in Eq. A.168,

- If the first case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.1.b.
- If the first case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.1.b.
- If the second case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.3.b.

- If the second case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.3.b.
- If the third case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.3.b.
- If the forth case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.3.b.
- If the fifth case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.5.b.
- If the fifth case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.5.b.
- If the sixth case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.5.b.
- If the seventh case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.7.b.
- If the seventh case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.7.b.

[**R6 is played by** w_j**,** $j > 1$] Eq. A.174 then implies

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\mathrm{R6}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

and, for all $1 < i \neq j \leq k$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

From Eq. A.173 we also have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = 0.$$

Here ρ_v, $>_v$ and $>_{v,w}$ are specified exactly as in the previous case.

[**R7 and R9 are played**] Those roles are played by two of the parents, while all other parents play role R11.

[**R7 is played by** w_1**, R9 is played by** w_j**,** $j > 1$] From Eqs. A.173 and A.174 we then have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = \varphi(\mathrm{R7}, \overline{x}_v^{w_1}, \overline{x}_{w_1}),$$

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\mathrm{R9}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

and ,for all $1 < i \neq j \leq k$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

Considering now the specification of the function φ in Eq. A.169,

- If the first case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.1.a.
- If the first case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.1.a.
- If the second case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.3.a.

- If the third case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.3.a.
- If the forth case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.5.a.
- If the fifth case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.5.a.
- If the sixth case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.7.a.
- If the sixth case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.7.a.

[**R**9 **is played by** w_1**,** **R**7 **is played by** w_j**,** $j > 1$] From Eqs. A.173 and A.174 we then have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = \varphi(\mathrm{R9}, \overline{x}_v^{w_1}, \overline{x}_{w_1}),$$

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\mathrm{R7}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

and, for all $1 < i \neq j \leq k$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

Here ρ_v, $>_v$ and $>_{v,w}$ are specified exactly as in the previous case.

[**R**7 **is played by** w_j**,** **R**9 **is played by** w_t**,** $j \neq t$**,** $j, t > 1$] From Eqs. A.173 and A.174 we have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = 0,$$

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\mathrm{R7}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

$$\varphi_{x_v^{w_t}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_t}) = \varphi(\mathrm{R9}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_t}),$$

and, for all $1 < i \leq k$, such that $i \notin \{j, t\}$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

Then, ρ_v, $>_v$ and $>_{v,w}$ are specified exactly as in the two previous cases.

[**R**8 **and** **R**10 **are played**] Those roles are played by two of the parents, while all other parents play role R11.

[**R**8 **is played by** w_1**,** **R**10 **is played by** w_j**,** $j > 1$] From Eqs. A.173 and A.174 we then have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = \varphi(\mathrm{R8}, \overline{x}_v^{w_1}, \overline{x}_{w_1}),$$

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\mathrm{R10}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

and, for all $1 < i \neq j \leq k$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

Considering now the specification of the function φ in Eq. A.170,

- If the first case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.1.b.

- If the first case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.1.b.
- If the second case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.3.b.
- If the third case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.3.b.
- If the forth case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.5.b.
- If the fifth case holds, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.5.b.
- If the sixth case holds, and the minimum is obtained at the first expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case III.7.b.
- If the sixth case holds, and the minimum is obtained at the second expression, then ρ_v, $>_v$ and $>_{v,w}$ are defined as in the proof of Theorem 26, case IV.7.b.

[**R**10 **is played by** w_1**,** **R**8 **is played by** w_j**,** $j > 1$] From Eqs. A.173 and A.174 we then have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = \varphi(\text{R10}, \overline{x}_v^{w_1}, \overline{x}_{w_1}),$$

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\text{R8}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

and, for all $1 < i \neq j \leq k$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

Here ρ_v, $>_v$ and $>_{v,w}$ are specified exactly as in the previous case.

[**R**8 **is played by** w_j**,** **R**10 **is played by** w_t**,** $j \neq t$**,** $j, t > 1$] From Eqs. A.173 and A.174 we have

$$\varphi_{x_v^{w_1}}(\overline{x}_v^{w_1}, \overline{x}_{w_1}) = 0,$$

$$\varphi_{x_v^{w_j}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_j}) = \varphi(\text{R8}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_j}),$$

$$\varphi_{x_v^{w_t}}([\![\mathbf{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, \eta]\!], [\![\mathbf{S}', \#'_{\mathsf{w}}, \#'_{\mathsf{b}}, \eta']\!], \sigma_{w_t}) = \varphi(\text{R10}, [\![\mathbf{S}, \#_{\mathsf{w}} - \#'_{\mathsf{w}}, \#_{\mathsf{b}} - \#'_{\mathsf{b}}, \eta]\!], \sigma_{w_t}),$$

and, for all $1 < i \leq k$, such that $i \notin \{j, t\}$,

$$\varphi_{x_v^{w_i}}(\overline{x}_v^{w_i}, \overline{x}_v^{w_{i-1}}, \overline{x}_{w_i}) = 0.$$

Then, ρ_v, $>_v$ and $>_{v,w}$ are specified exactly as in the two previous cases.

Until now, for each variable $v \in V$, we have specified the action sequence ρ_v and the order $>_v$ over the elements of ρ_v. For each $w \in \mathsf{pred}(v)$, we have specified the order $>_{v,w}$, and proved that all $>_v \cup >_{v,w}$ and $>_w \cup >_{v,w}$ form strict partial orders over their domains. Similarly to the uniform cost case, this construction allows us to apply now Theorem 22 on the (considered as sets) sequences ρ_v and orders $>_v$ and $>_{v,w}$, proving that

$$>= \bigcup_{v \in V} (>_v \cup \bigcup_{w \in \mathsf{pred}(v)} >_{v,w})$$

forms a strict partial order over the union of $\rho_{v_1}, \ldots, \rho_{v_n}$.

Finally, we note that the plan extraction step of the algorithm polytree-1-dep corresponds exactly to the above construction along Eqs. A.52-A.57, A.59-A.61, A.63-A.64, A.71-A.72, A.74-A.75, A.83-A.84, A.86, A.88, A.95-A.96, A.98, A.100, A.107-A.108, A.110-A.111, A.113, A.120-A.121, A.123, A.131-A.132, A.134-A.135, A.137, A.144-A.145, A.147-A.148, A.156, providing us in poly-time with concrete cost-optimal plan corresponding to the optimal solution for COP_Π.

(II) We now prove that if Π is solvable, then we must have $\alpha < \infty$. Assume to the contrary that this is not the case. Let Π be a solvable $\mathbf{P}(1)$ task, and let (using Theorem 26) ρ be an irreducible, post-3/2 plan for Π. Given such ρ, let a COP assignment $\overline{x}_\rho$ be defined as follows.

1. For each COP variable x_v, the assignment $\overline{x}_\rho$ provides the value $\sigma_v \in \trianglerighteq^*[\sigma(v)]$ such that $|\sigma_v| = |\rho{\downarrow_v}| + 1$.

2. For each variable $v \in V$, such that $\mathsf{pred}(v) \neq \emptyset$, find the (at most two) parents that prevail the actions in $\rho[v]$. Let w be such a parent that performs a role $\mathrm{R} \in \{\mathrm{R1}, \mathrm{R2}, \mathrm{R3}, \mathrm{R5}, \mathrm{R6}, \mathrm{R7}, \mathrm{R8}\}$, and w' be the other such parent that performs one of the roles $\mathrm{R}' \in \{\mathrm{R4}, \mathrm{R9}, \mathrm{R10}, \mathrm{R11}\}$. (By definition of post-3/2 action sequences, the rest of the parents all perform role R11.) Given that, if $|\mathsf{pred}(v)| = k > 0$, we adopt an ordering of $\mathsf{pred}(v)$ such that $w_1 = w$ and $w_k = w'$. First, the assignment $\overline{x}_v^{w_k}$ to COP variable $x_v^{w_k}$ provides the value $[\![\mathrm{S1}, \lceil \frac{|\sigma_v|-1}{2} \rceil, \lfloor \frac{|\sigma_v|-1}{2} \rfloor, |\sigma_v| - 1]\!]$. Then, for $1 \leq i < k$, the assignment $\overline{x}_v^{w_i}$ to COP variable $x_v^{w_i}$ provides the value $[\![\mathrm{S}, \#_{\mathsf{w}}, \#_{\mathsf{b}}, |\sigma_v| - 1]\!]$, where

$$\mathrm{S} = \begin{cases} \mathrm{S2}, & \mathrm{R}' = \mathrm{R4} \\ \mathrm{S4}, & \mathrm{R}' = \mathrm{R9} \\ \mathrm{S5}, & \mathrm{R}' = \mathrm{R10} \\ \mathrm{S1}, & \mathrm{R}' = \mathrm{R11} \end{cases}$$

and $\#_{\mathsf{w}}$ and $\#_{\mathsf{b}}$ are the numbers of actions in $\rho{\downarrow_v}$ that change the value of v to w_v and b_v, respectively, while being prevailed by the value of w_1.

From Eq. A.160-A.174 we then have that, for each $v \in V$, if $\mathsf{pred}(v) = \emptyset$, then $\varphi_{x_v}(\overline{x}_v) = cost(\rho{\downarrow_v})$. Otherwise, if $\mathsf{pred}(v) = \{w_1, \ldots, w_k\}$, then $\varphi_{x_v}(\overline{x}_v, \overline{x}_v^{w_k}) = 0$, and $\sum_{w \in \mathsf{pred}(v)} \varphi_{x_v^w}(\overline{x}_\rho) = cost(\rho{\downarrow_v})$. Therefore, we have

$$\sum_{\varphi \in \mathcal{F}} \varphi(\overline{x}_\rho) \;\; = \;\; \sum_{v \in V} cost(\rho{\downarrow_v}) \;\; = \;\; cost(\rho),$$

which is what we had to prove. ∎

Theorem 27 *Cost-optimal planning for* $\mathbf{P}(1)$ *is tractable.*

Proof: The correctness of the polytree-1-dep algorithm is given by Lemma 11. We now show that, given a planning task $\Pi \in \mathbf{P}(1)$, the corresponding constraint optimization problem COP_Π can be constructed and solved in time polynomial in the description size of Π.

Let n be the number of state variables in Π. In polytree-1-dep-uniform, we first construct the constraint optimization problem COP_Π over $\Theta(n^2)$ variables $\mathcal{X}$ with domain sizes being bounded either by $O(n)$ or by $O(n^3)$ (for COP variables representing state variables and causal graph edges, respectively). The number of functional components in COP_Π is $\Theta(n^2)$, each defined over one variable with domain size of $O(n)$ and either one or two variables with domain sizes of $O(n^3)$. The construction is linear in the size of the resulting COP, and thus is accomplished in time $O(n^9)$.

Applying then to COP_Π a tree-decomposition that clusters the scopes of the functional components $\mathcal{F}$, we arrive into an equivalent, tree-structured constraint optimization problem over $\Theta(n^2)$ variables with domains of size $O(n^7)$. Such a tree-structured COP can be solved in time $O(xy^2)$ where x is the number of variables and y is an upper bound on the size of a variable's domain (Dechter, 2003). Therefore, solving COP_Π can be done in time $O(n^{16})$. As this dominates both the time complexity of constructing COP_Π, and the time complexity of extracting a plan from the optimal solution to COP_Π (see the proof of (I) in Lemma 11), the overall complexity of the algorithm polytree-1-dep-uniform is $O(n^{16})$, and therefore polynomial in the description size of Π. ■

Appendix B

Experimental Evaluation in Detail

B.1 Fork-Decompositon

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
airport-ipc4																	
01	8	10	0.01	9	0.00	9	0.00	9	0.00	9	0.00	9	0.72	11	0.00	9	0.00
02	9	12	0.03	15	0.01	15	0.03	10	0.00	10	0.00	10	1.23	13	0.00	10	0.00
03	17	86	0.25	133	0.07	93	0.31	18	0.04	18	0.03	29	5.10	164	0.00	57	0.00
04	20	22	0.02	21	0.02	21	0.02	21	0.02	21	0.01	21	1.32	23	0.00	21	0.00
05	21	23	1.29	30	0.06	27	1.43	22	0.01	22	0.01	22	46.54	27	0.00	22	0.00
06	41	513	36.72	639	1.54	567	45.25	42	0.16	42	0.17	42	123.13	738	0.01	418	0.02
07	41	514	37.00	632	1.53	550	44.15	42	0.17	42	0.17	42	117.56	742	0.01	405	0.02
08	62			21544	166.51			24372	25.42	9623	1549.13	203	602.09	27032	0.28	9687	0.90
09	71							152408	64.92	89525	466.14	12956	993.07	175717	2.47	56484	7.62
10	18	19	0.02	19	0.02	19	0.03	19	0.02	19	0.01	19	2.45	21	0.00	19	0.00
11	21	23	1.90	30	0.08	27	2.13	22	0.02	22	0.01	22	65.36	27	0.00	22	0.01
12	39	475	54.18	728	2.76	568	71.23	40	0.21	40	0.21	40	169.02	873	0.01	392	0.03
13	37	434	47.48	663	2.60	479	59.82	38	0.20	38	0.21	38	134.87	822	0.01	342	0.03
14	60			25110	334.72			30637	51.23	8968	238.16	62	714.76	35384	0.39	9196	1.11
15	58			23317	307.60			28798	46.20	8931	267.81	59	647.05	33798	0.38	8200	1.01
16	79							1031524	200.95	305340	1077.90			1247467	19.72	221993	49.03
17	88															1043661	310.89
19	90															831632	253.21
21	101							7326	372.92	102	10.28			18809	0.42	3184	1.12
22	148							1119943	762.02							159967	105.29
36	109							34365	853.70					63061	1.44		
depots-ipc3																	
01	10	114	0.24	279	0.11	161	0.32	11	0.00	11	0.00	45	0.77	329	0.00	136	0.00
02	15	1134	10.82	9344	12.40	2638	22.68	738	3.24	16	1.14	898	11.56	15404	0.11	3771	0.17
03	27							348288	20.69	239313	222.35	103089	247.13	2930398	27.20	1204646	97.62
04	30							1284048	52.05	1273762	529.34						
07	21							211820	37.54			41328	324.19	6501100	71.58	1331701	166.76
10	24							3241083	157.52								
13	25							1427824	116.06								
driverlog-ipc3																	
01	7	49	0.05	37	0.01	37	0.04	8	0.04	8	0.03	44	0.47	182	0.00	20	0.00
02	19	15713	18.27	18452	10.29	15794	23.80	20	0.13	20	0.26	15998	4.55	68927	0.36	54283	0.52
03	12	164	0.25	190	0.13	163	0.31	13	0.16	13	0.25	863	1.25	16031	0.09	2498	0.03
04	16	6161	19.15	10778	17.14	7665	29.88	17	0.49	17	2.41	22933	12.20	999991	8.12	393673	6.56
05	18	13640	45.02	11400	18.91	10984	46.16	2614	0.60	19	4.58	24877	18.77	629080	361.57	1724611	34.73
06	11	608	5.21	795	3.60	492	6.05	291	1.35	12	9.72	3804	10.08	681757	7.64	54451	1.71
07	13	864	9.56	1730	7.71	1006	13.80	14	1.42	14	15.35	25801	41.34	6349767	81.53	493480	17.31
08	22							287823	7.34	2952	20.31						
09	22			198651	849.04			15504	1.70	23	10.43						
10	17	4304	199.81	16099	85.74	4037	200.52	18	1.64	18	18.54	18234	68.22				
11	19	43395	1421.90	41445	186.53	39069	1395.51	34137	1.99	10790	17.01	559623	1193.00			6141130	330.22
13	26							1298884	19.52	870875	35.33						

Table B.1: Runtimes of cost-optimal heuristic-search planners on the AIRPORT, DEPOTS, and DRIVERLOG domains. The description of the planners is given in Section 4.4.2; here the fork-decomposition heuristics are computed fully online. Column *task* denotes problem instance, column h^* denotes optimal solution length. Other columns capture the run *time* and number of expanded *nodes*.

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
blocks-ipc2																	
04-0	6	15	0.01	46	0.01	17	0.01	7	0.03	7	0.03	7	0.36	93	0.00	25	0.00
04-1	10	14	0.01	31	0.00	15	0.00	11	0.04	11	0.03	11	0.39	66	0.00	23	0.00
04-2	6	7	0.01	26	0.00	10	0.00	7	0.04	7	0.03	7	0.38	63	0.00	18	0.00
05-0	12	32	0.03	302	0.06	113	0.08	13	0.30	13	0.96	13	1.32	467	0.00	145	0.00
05-1	10	37	0.03	280	0.06	98	0.07	11	0.29	11	0.96	11	1.36	567	0.00	135	0.00
05-2	16	152	0.09	596	0.10	348	0.18	17	0.29	17	0.95	17	1.49	792	0.00	297	0.00
06-0	12	33	0.04	766	0.27	207	0.25	13	0.95	13	8.56	13	4.10	1826	0.00	276	0.00
06-1	10	41	0.07	2395	0.74	578	0.78	11	0.90	11	8.34	11	4.17	4887	0.01	755	0.01
06-2	20	855	0.80	5444	1.23	3352	2.88	733	0.87	85	8.84	31	4.29	6385	0.02	2556	0.03
07-0	20	278	0.56	20183	8.26	4022	8.18	577	1.93	144	23.32	22	11.47	37157	0.14	5943	0.11
07-1	22	6910	11.22	59207	17.37	38539	49.71	10071	1.70	1835	21.05	174	11.25	63376	0.21	33194	0.46
07-2	20	1458	2.85	46009	15.05	18854	29.61	1855	1.59	782	20.37	90	10.99	55218	0.19	18293	0.29
08-0	18	1533	4.79	344157	179.42	69830	208.07	5557	3.67	678	36.80	25	26.00	519107	2.28	94671	2.07
08-1	20	10040	27.97	517514	236.64	191352	475.33	45711	3.88	11827	33.49	151	26.57	636498	2.60	199901	3.85
08-2	16	479	1.79	237140	136.18	32567	110.76	277	3.63	54	32.53	17	25.85	433144	1.93	52717	1.30
09-0	30							1233374	16.00	971409	77.74	464	56.76	798464	936.76	3840589	85.00
09-1	28	3435	18.17					95068	7.35	58873	63.15	82	56.98	591457	229.73	1200345	32.06
09-2	26	6379	35.22					161719	13.54	20050	82.45	81	57.02	596316	030.02	1211463	32.15
10-0	34											1800	114.26				
10-1	32									12063665	228.76	1835	115.19				
10-2	34											3685	116.75				
11-0	32									7046739	141.44	2678	213.32				
11-1	30											1510	203.79				
11-2	34											3984	213.97				
12-0	34											1184	370.06				
12-1	34											614	382.34				
13-0	42											83996	860.45				
13-1	44											163438	1104.27				
14-0	38											2779	1063.02				
14-1	36											7154	1087.40				
grid-ipc1																	
01	14	571	60.28	1117	9.49	472	55.87	660	8.63	467	121.10			6446	0.08	190	0.10
02	26							3392724	50.35	3244132	241.94					664016	231.26
gripper-ipc1																	
01	11	214	0.04	240	0.02	214	0.05	12	0.00	12	0.00	33	0.11	236	0.00	208	0.00
02	17	1768	0.54	1832	0.36	1803	0.75	18	0.11	18	0.08	680	0.37	1826	0.01	1760	0.01
03	23	11626	5.38	11736	4.05	11689	8.11	11514	0.47	2094	1.75	7370	1.52	11736	0.04	11616	0.08
04	29	68380	43.58	68558	35.24	68479	70.72	68380	1.24	68190	8.05	55568	10.29	68558	0.27	68368	0.56
05	35	376510	328.10	376784	296.59	376653	560.93	376510	3.52	376510	19.46	344386	79.96	376772	1.59	376496	3.51
06	41							1982032	13.42	1982032	42.16	1911592	577.49	1982394	9.59	1982016	21.57
07	47							10091986	61.66	10091986	106.84			10092464	51.10	10091968	119.64
freecell-ipc3																	
01	8	234	1.54	974	4.88	274	3.25	87	3.12	9	38.74	9	13.01	3437	0.03	1043	0.15
02	14	30960	107.07	75150	230.54	37131	224.62	31487	40.40			466	70.29	130883	1.46	41864	10.77
03	18	197647	877.16					95805	140.96			1589	169.39	944843	11.45	210503	75.62
04	26							943074	86.78			15848	341.02	302132	638.80	600525	247.70
05	30							5950977	243.74			40642	916.44			1408035	1062.25
logistics-ipc1																	
01	26							1918881	41.03	949586	34.82	211955	1700.26				
05	22	3293	945.35					768161	18.69	609393	35.27						
31	13	436	9.67	1981	2.53	1284	21.84	494	0.42	14	2.11	481	6.58	155645	1.66	32282	0.57
32	20	392	2.57	2704	2.24	962	5.53	21	0.16	21	0.72	9598	7.08	245325	2.07	81156	1.00
33	27									529338	32.55						
logistics-ipc2																	
04-0	20	21	0.02	193	0.06	65	0.06	21	0.03	21	0.05	21	0.34	11246	0.05	4884	0.03
04-1	19	20	0.03	570	0.13	293	0.16	20	0.03	20	0.04	20	0.37	9249	0.04	4185	0.03
04-2	15	16	0.02	117	0.03	79	0.05	16	0.04	16	0.05	16	0.36	4955	0.02	1205	0.01
05-0	27	28	0.05	2550	0.98	1171	1.09	28	0.10	28	0.38	28	0.58	109525	0.64	74694	0.59
05-1	17	18	0.03	675	0.19	427	0.31	18	0.10	18	0.38	18	0.72	22307	0.13	6199	0.05
05-2	8	9	0.02	24	0.01	13	0.02	9	0.09	9	0.38	9	0.78	1031	0.00	280	0.00
06-0	25	26	0.06	4249	1.85	2461	2.54	26	0.18	26	1.23	26	1.03	490207	3.40	202229	1.92
06-1	14	15	0.03	181	0.09	99	0.13	15	0.18	15	1.26	15	1.16	24881	0.16	3604	0.03
06-2	25	26	0.05	2752	1.22	1394	1.51	26	0.19	26	1.26	26	1.03	476661	3.32	200012	1.98
06-9	24	25	0.04	2395	0.94	1428	1.34	25	0.18	25	1.22	25	1.02	422557	2.95	133521	1.29
07-0	36	37	0.42	251287	203.64	98053	386.80	525	0.65	37	4.87	24317	35.46				
07-1	44	1689	10.08					666324	8.83	49	4.94	362179	453.06				
08-0	31	32	0.42	82476	78.73	35805	161.33	1042	0.96	32	6.90	14890	33.50				
08-1	44	45	0.66	1183608	1306.92			16708	1.15	45	7.21	114155	198.84				
09-0	36	37	0.54	351538	407.06	167038	883.68	20950	1.56	37	9.46	32017	83.16				
09-1	30	31	0.50	59336	80.88	25359	168.73	31	1.27	31	9.43	6720	26.48				
10-0	45	46	2.26							668834	29.73						
10-1	42	43	2.10							1457130	43.00						
11-0	48	697	26.78							701106	37.42						
11-1	60	21959	696.23														
12-0	42	43	2.78							775996	43.56						
12-1	68									2222340	87.47						

Table B.2: Similar to Table B.1 for the Blocksworld, Grid, Gripper, Freecell, Logistics-ipc1, and Logistics-ipc2 domains.

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
miconic-strips-ipc2																	
01-0	4	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.01	5	0.00	5	0.00
01-1	3	5	0.00	5	0.00	5	0.00	4	0.00	4	0.00	4	0.00	5	0.00	4	0.00
01-2	4	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.01	5	0.00	5	0.00
01-3	4	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00
01-4	4	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.01	5	0.00	5	0.00
02-0	7	19	0.00	22	0.00	19	0.00	8	0.00	8	0.00	26	0.01	30	0.00	20	0.00
02-1	7	21	0.00	23	0.00	21	0.00	8	0.00	8	0.00	26	0.01	30	0.00	22	0.00
02-2	7	21	0.00	23	0.00	21	0.00	8	0.00	8	0.00	27	0.00	30	0.00	22	0.00
02-3	7	24	0.01	24	0.00	24	0.00	8	0.00	8	0.00	20	0.01	26	0.00	17	0.00
02-4	7	19	0.00	22	0.00	19	0.00	8	0.00	8	0.00	23	0.01	31	0.00	20	0.00
03-0	10	86	0.01	129	0.01	98	0.01	11	0.00	11	0.00	100	0.03	193	0.00	105	0.00
03-1	11	120	0.01	168	0.01	147	0.01	12	0.00	12	0.00	140	0.02	218	0.00	150	0.00
03-2	10	137	0.01	143	0.01	137	0.01	11	0.00	11	0.00	122	0.02	164	0.00	92	0.00
03-3	10	96	0.01	153	0.01	117	0.01	11	0.00	11	0.00	131	0.02	197	0.00	130	0.00
03-4	10	103	0.01	149	0.01	115	0.01	11	0.00	11	0.00	114	0.02	190	0.00	114	0.00
04-0	14	524	0.06	843	0.08	686	0.12	15	0.01	15	0.01	669	0.10	1182	0.00	866	0.00
04-1	13	505	0.06	817	0.08	663	0.12	14	0.01	14	0.01	634	0.11	1176	0.00	860	0.00
04-2	15	685	0.08	942	0.09	802	0.13	16	0.01	16	0.01	822	0.12	1277	0.00	969	0.00
04-3	15	681	0.07	942	0.09	798	0.13	16	0.01	16	0.01	820	0.12	1319	0.00	970	0.00
04-4	15	685	0.07	942	0.09	802	0.13	16	0.01	16	0.01	821	0.12	1334	0.00	969	0.00
05-0	17	2468	0.37	4009	0.66	3307	0.93	18	0.06	18	0.05	2829	0.44	6350	0.03	4387	0.03
05-1	17	2807	0.42	4345	0.71	3677	1.01	18	0.06	18	0.05	3260	0.49	6602	0.03	4664	0.03
05-2	15	1596	0.29	2981	0.55	2275	0.73	16	0.06	16	0.05	1594	0.32	5565	0.03	3524	0.03
05-3	17	2256	0.36	3799	0.62	3104	0.87	18	0.06	18	0.05	2568	0.42	5944	0.03	4140	0.03
05-4	18	3210	0.46	4732	0.78	4267	1.11	19	0.06	19	0.05	3953	0.55	6949	0.04	5268	0.04
06-0	19	9379	1.98	17665	4.74	13531	5.90	20	0.18	20	0.32	9312	1.76	30786	0.20	21194	0.20
06-1	19	9106	1.93	18134	4.75	14052	5.94	20	0.18	20	0.32	10252	1.96	30093	0.20	21255	0.20
06-2	20	10900	2.19	19084	4.90	15111	6.28	21	0.18	21	0.32	11247	2.11	32390	0.21	21694	0.21
06-3	20	12127	2.43	21708	5.69	17807	7.19	21	0.17	21	0.32	14216	2.56	32574	0.21	24552	0.23
06-4	21	13784	2.62	23255	5.93	19536	7.66	22	0.17	22	0.32	16880	3.04	33793	0.22	26167	0.24
07-0	23	53662	13.29	96092	37.56	79449	46.76	24	0.32	24	1.75	56686	14.31	155466	1.22	116685	1.32
07-1	24	56328	13.86	99109	38.56	83677	47.49	7001	0.38	25	1.75	63035	16.33	164470	1.29	118494	1.33
07-2	22	48141	12.52	96139	38.02	78471	46.17	1646	0.33	23	1.71	55751	13.98	161342	1.27	119688	1.36
07-3	22	46867	12.11	93117	36.63	75424	44.43	1861	0.33	23	1.74	53121	13.27	155176	1.23	114649	1.30
07-4	25	84250	18.24	126595	46.11	111984	61.34	23159	0.52	26	1.71	96327	24.76	168219	1.33	140128	1.58
08-0	27	272580	81.51	485051	267.27	408114	317.78	41629	0.91	28	4.18	290649	104.18	755255	7.16	594032	7.95
08-1	27	284415	86.93	527216	288.07	446837	347.43	42679	0.90	28	4.25	339177	123.10	794365	7.56	636587	8.66
08-2	26	207931	66.37	414294	235.89	330993	271.03	37744	0.86	27	4.25	204614	73.39	731622	6.92	534711	7.37
08-3	28	369479	104.29	598031	320.33	527216	392.87	140453	1.94	29	4.21	435617	160.49	833421	7.97	690267	9.29
08-4	27	297516	87.65	507910	278.64	431432	333.91	62933	1.16	28	4.12	315339	111.84	771608	7.33	613253	8.43
09-0	31	1461729	497.72					684737	9.07	126918	8.89	1555286	794.93	3685552	41.04	3006991	49.12
09-1	30	1207894	438.69	2335166	1787.13			406041	5.61	100937	8.73	1344815	683.05	3649801	40.32	2893803	47.54
09-2	30	1294691	460.11	2340411	1791.16			442547	6.06	82946	8.63	1357681	692.11	3576134	39.61	2895182	47.26
09-3	32	1840936	589.09					765455	10.00	277302	11.14	2083168	1051.95	3796035	42.13	3304570	53.29
09-4	28	1252484	467.94					317692	4.65	29	7.03	1231554	605.01	3589382	39.29	2956995	48.84
10-0	33							2436164	35.24	863244	23.76			15804498	200.90	13267920	250.58
10-1	32							2340169	34.09	335745	15.68			16472633	208.39	13720664	256.89
10-2	32							1735477	25.29	486286	17.72			15867374	201.01	12497087	236.89
10-3	34							3952148	55.86	940556	24.24			16309701	208.42	13801989	262.53
10-4	33							2715866	39.44	625559	19.91			16472551	209.13	13925654	262.57
11-0	37							11473359	183.60	4724980	93.56						
11-1	34							7535468	124.80	1934943	47.91						
11-2	38							14645785	233.68	6330198	120.71						
11-3	38									5809711	110.10						
11-4	35							5853546	95.56	1082086	32.22						
mprime-ipc1																	
01	5	196	0.19	10	0.03	24	0.07	6	2.00	6	20.45	108	49.59	3636	0.07	68	0.04
02	7	11604	422.83	44045	1620.68	2565	242.83	3317	88.58							12606	36.65
03	4	427	35.09	7	0.50	11	3.15	36	33.64	5	463.85			9868	0.67	5	0.07
04	8	3836	6.62	1775	1.17	1093	3.44	9	6.09	9	82.71	19076	781.74	599590	23.58	200	0.24
05	11							1705009	127.53							1488157	1638.78
07	5	3314	14.91	47	0.15	346	3.07	1667	46.72					18744	0.56	11	0.04
08	6							1469752	403.45							7650	84.33
09	8	19838	454.91	100188	1798.69	5227	284.13	21993	36.25					2197646	71.69	19023	30.26
11	7	9	0.16	219	0.54	8	0.16	8	4.69	8	62.68	22	394.26	73260	2.21	915	0.54
12	6	16320	192.10	8118	46.69	5243	95.01	34763	11.45	42055	143.27	25665	724.12	108652	3.50	1520	1.78
15	6															1039	178.55
16	6	252	171.97			448	447.49	473	81.42					425144	32.17	7962	35.65
17	4			453	671.03									172736	42.48	5	1.06
19	6							123039	313.25							36013	533.75
21	6													1503293	103.23	15250	101.75
25	4	75	0.10	30	0.04	29	0.08	5	0.48	5	2.75	85	8.71	383	0.00	6	0.00
26	6							172432	46.33	189154	454.69			819590	61.01	440	2.69
27	5	54	2.28	1772	33.82	9	1.31	6	11.59	6	154.43			84079	3.50	831	2.08
28	7	8	0.03	403	0.23	37	0.08	8	1.88	8	22.55	128	146.80	17333	0.25	211	0.06
29	4	182	4.53	56	1.11	32	1.79	5	14.92	5	201.40			3187	0.17	7	0.10
31	4	248	52.86	46	7.83	19	11.79	419	99.87					3584	0.19	11	0.17
32	7	31759	133.33	12436	34.94	11839	95.52	19429	21.61	7269	292.37	11073	1701.00	115479	2.75	3096	1.74
34	4	234	11.65	46	2.13	23	3.08	450	151.69					3618	0.19	11	0.18
35	5	392	3.09	290	2.54	84	1.89	359	3.63	6	43.43	706	96.55	2476	0.05	44	0.03

Table B.3: Similar to Table B.1 for the Miconic and Mprime domains.

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
mystery-ipc1																	
01	5	7	0.01	6	0.00	6	0.01	6	0.20	6	1.79	10	5.38	30	0.00	8	0.00
02	7	2404	64.94	8012	234.10	722	47.50	1672	82.70					770852	21.85	2368	4.47
03	4	73	1.92	7	0.12	11	0.59	5	16.46	5	193.75	65	811.87	507	0.02	5	0.03
04	∞	0	0.01			0	0.00										
07	∞	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
09	8	3049	47.68	10764	137.61	1215	40.75	3165	29.34			3868	670.08	138289	2.18	1458	1.44
11	7	9	0.02	33	0.03	8	0.02	8	1.51	8	16.59	34	41.20	426	0.00	19	0.00
12	∞			2093419	938.05			2102777	14.61	2102729	27.84			2102777	15.09	1177842	21.87
15	6													279973	13.21	135	2.62
16	∞	0	0.14			0	0.19										
17	4	354	200.98	85	26.31	83	90.17	198	445.85					5400	0.41	5	0.35
18	∞	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
19	6			4968	183.24			12478	96.38					133871	3.65	1516	5.44
20	7							285069	59.22	547246	578.39			686125	23.28	718	3.76
24	∞	0	0.13			0	0.30										
25	4	9	0.02	10	0.01	9	0.02	5	0.10	5	0.10	14	1.22	31	0.00	6	0.00
26	6	1807	50.40	1835	25.34	1344	60.20	2526	5.94	346	70.78	3107	291.36	8455	0.10	37	0.05
27	5	14	0.27	159	1.61	6	0.22	6	4.80	6	80.48	7	243.78	2174	0.03	73	0.04
28	7	8	0.01	47	0.02	15	0.02	8	0.63	8	6.77	31	16.67	843	0.00	32	0.00
29	4	31	0.26	14	0.10	10	0.17	5	8.94	5	107.10	27	536.30	153	0.01	7	0.02
30	9							42112	28.07	44893	357.07			1977063	38.26	26686	28.27
openstacks-ipc5																	
01	23	2264	0.49	3895	1.19	3070	1.36	24	0.05	24	0.06	2000	1.02	4822	0.01	4016	0.03
02	23	2617	0.56	4485	1.32	3561	1.57	24	0.06	24	0.06	2378	1.07	5501	0.02	4594	0.04
03	23	2264	0.49	3895	1.15	3070	1.36	24	0.06	24	0.06	2000	1.02	4822	0.01	4016	0.03
04	23	2264	0.49	3895	1.15	3070	1.36	24	0.06	24	0.06	2000	1.02	4822	0.02	4016	0.03
05	23	2264	0.48	3895	1.15	3070	1.35	24	0.06	24	0.05	2000	1.02	4822	0.01	4016	0.03
06	45	366768	255.00	779710	1599.86	587482	1498.20	621008	4.85	279614	7.86	379735	217.37	882874	4.91	822514	18.71
07	46	410728	277.99	760668	1546.44	606782	1515.46	594758	4.69	264535	7.34	405564	226.32	836647	4.62	787163	17.81
pathways-ipc5																	
01	6	1624	0.03	1299	0.02	1299	0.03	7	1.14	7	0.79	1405	0.28	1624	0.00	36	0.00
02	12	2755	0.08	2307	0.06	2437	0.09	1946	2.56	13	42.11	990	0.29	2984	0.02	348	0.01
03	18	44928	2.59	20416	1.06	29106	2.14	21671	6.43	14901	129.23	14772	6.99	87189	1.06	4346	0.16
04	17	126950	11.45	33788	2.97	58738	7.07			98484	288.39	34206	27.00	456143	8.22	104068	2.61
pipesworld-notankage-ipc4																	
01	5	121	0.15	109	0.05	121	0.18	6	0.04	6	0.04	6	2.79	121	0.00	13	0.00
02	12	1413	2.05	1542	0.86	1413	2.42	169	0.30	13	0.17	435	3.07	1808	0.01	792	0.02
03	8	1742	5.26	3001	3.31	1742	6.43	9	1.15	9	0.69	128	3.84	3293	0.02	262	0.02
04	11	7007	24.71	8911	12.43	7007	30.79	651	1.95	12	7.05	812	8.84	16088	0.11	2925	0.13
05	8	4093	27.45	6805	19.74	4093	35.40	77	5.63	9	21.15	155	16.53	11128	0.12	1121	0.15
06	10	12401	105.37	27377	103.75	12401	140.53	1299	5.26	61	39.31	1151	23.41	49905	0.48	7102	0.72
07	8	4370	71.75	9168	68.10	4370	105.53	233	19.78	9	59.70	185	29.88	46502	0.57	2631	0.48
08	10	18851	406.67	56189	483.28	20584	600.94	561	12.42	497	94.69	1673	48.84	273585	3.39	22874	3.58
09	13							104875	25.48			10478	74.26	5513309	80.62	321861	68.99
10	18							2982520	66.89			689832	1439.64			11121245	1579.77
11	20			472950	1577.22			90598	9.20	52159	43.24	108503	625.52	710123	3.86	107061	14.51
12	24							594661	12.41	416184	109.43	433296	1117.57	2467804	13.83	464982	56.82
13	16			117475	899.72			12835	34.28			24224	1019.65	481045	3.14	33417	6.38
14	30							13255718	119.54								
15	26							648132	65.43					4921698	34.90	555619	105.49
17	22							3200672	90.07								
19	24							8767431	150.88								
21	14	23833	1663.46	49035	495.53			3992	18.13	948	159.63			157782	1.31	8966	2.42
23	18							296506	49.11	104750	256.13					481859	229.00
24	24							7315150	142.82								
41	12															114257	250.18
pipesworld-tankage-ipc4																	
01	5	77	0.13	126	0.07	105	0.20	6	3.54	6	0.13	6	3.88	128	0.00	13	0.01
02	12	960	1.20	1005	0.60	960	1.55	110	3.04	13	0.20	179	6.04	1012	0.01	659	0.02
03	8	20803	155.53	52139	158.91	20803	207.57	244	22.64	9	36.89	818	24.47	52983	0.77	1802	1.33
04	11	110284	1004.10	157722	668.67	110284	1408.50	3892	16.68	12	155.03	8116	64.68	221429	3.06	41540	14.49
05	8	6531	73.63	13148	79.04	6531	112.61	376	15.46	9	120.06	313	59.99	12764	0.21	2834	1.61
06	10	20171	329.40	43583	310.24	20171	460.45	1794	328.18	11	201.44	3102	97.31	58487	0.87	15746	6.61
07	8											2695	339.76	5404036	198.08	104531	420.47
08	11							96043	191.77								
11	22							660104	28.60	660102	162.93			4116344	30.67	752867	334.42
13	16							188517	122.11								
15	30							2546587	141.12								
17	44							12850247	352.46								
21	14							13241	69.80					4423951	65.44	126845	222.23
31	39							1357801	124.64					1726598	13.56	919764	381.66
trucks-ipc5																	
01	13	1691	0.41	1027	0.22	1039	0.40	14	0.03	14	0.02	285	0.56	5774	0.02	402	0.01
02	17	9624	2.68	2898	0.57	2957	1.35	4192	0.22	18	0.17	1413	1.04	28348	0.14	939	0.03
03	20	80693	71.37	20752	19.93	22236	31.25	199405	2.89	173790	6.88	4049	4.43	379582	2.97	9465	0.40
04	23	1753866	1237.60	1205793	850.34	1315672	1394.88	2591561	29.17	2568634	56.96	8817	7.75	2990366	26.65	209140	9.43
05	25							23444940	392.99			14744	23.12			1248571	90.78
06	30											308920	343.47				
07	23	2134728	1313.60	719751	408.75	755608	820.55	7575415	88.91	8080496	117.13	43270	27.62	12410588	117.92	223011	19.34
08	25											49663	47.61			3106944	403.36
09	28											233577	248.21				

Table B.4: Similar to Table B.1 for the Mystery, Openstacks, Pathways, Pipesworld-NoTankage, Pipesworld-Tankage, and Trucks domains.

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
psr-small-ipc4																	
01	8	10	0.00	10	0.00	10	0.00	9	0.00	9	0.00	9	0.01	11	0.00	9	0.00
02	11	52	0.01	55	0.00	52	0.01	12	0.00	12	0.00	20	0.08	71	0.00	47	0.00
03	11	31	0.01	31	0.00	31	0.00	12	0.00	12	0.00	20	0.04	33	0.00	28	0.00
04	10	66	0.04	91	0.03	73	0.06	11	0.00	11	0.00	12	0.34	332	0.00	102	0.00
05	11	75	0.01	79	0.01	75	0.02	12	0.00	12	0.00	23	0.11	154	0.00	69	0.00
06	8	10	0.00	10	0.00	10	0.00	9	0.00	9	0.00	9	0.01	11	0.00	9	0.00
07	11	61	0.01	61	0.00	61	0.01	12	0.00	12	0.00	26	0.09	122	0.00	62	0.00
08	8	24	0.01	29	0.00	25	0.01	9	0.00	9	0.00	9	0.12	128	0.00	52	0.00
09	8	18	0.01	19	0.00	18	0.00	9	0.00	9	0.00	9	0.06	49	0.00	20	0.00
10	7	131	0.20	183	0.18	155	0.32	8	0.04	8	0.04	18	1.04	1358	0.00	376	0.01
11	19	149	0.03	149	0.02	149	0.04	20	0.00	20	0.00	96	0.19	153	0.00	142	0.00
12	16	120	0.03	123	0.02	120	0.04	17	0.00	17	0.00	40	0.17	153	0.00	113	0.00
13	15	90	0.02	90	0.01	90	0.02	16	0.00	16	0.00	59	0.16	95	0.00	86	0.00
14	9	19	0.00	19	0.00	19	0.00	10	0.00	10	0.00	13	0.06	27	0.00	18	0.00
15	10	1200	6.55	708	6.25	769	9.91	11	0.46	11	2.58	356	18.99	3562	0.02	324	0.02
16	25	2328	0.65	2158	0.34	2176	0.85	975	0.11	26	0.12	2287	1.34	2742	0.01	1876	0.01
17	9	15	0.00	15	0.00	15	0.00	10	0.00	10	0.00	13	0.03	16	0.00	14	0.00
18	12	85	0.03	90	0.01	85	0.03	13	0.00	13	0.00	29	0.21	158	0.00	91	0.00
19	25	8025	4.31	7856	2.19	7876	5.80	2910	0.27	26	0.77	6338	4.46	9009	0.04	6925	0.08
20	17	80	0.02	80	0.01	80	0.02	18	0.00	18	0.00	52	0.18	84	0.00	75	0.00
21	10	28	0.01	28	0.00	28	0.01	11	0.00	11	0.00	21	0.12	42	0.00	31	0.00
22	33	163299	405.65	176058	245.42	168685	617.45	34	0.28	34	0.87	22315	8.16	189516	0.67	177138	1.43
23	12	77	0.04	93	0.03	77	0.06	13	0.00	13	0.01	30	0.43	200	0.00	116	0.00
24	10	28	0.01	28	0.00	28	0.01	11	0.00	11	0.00	21	0.12	42	0.00	31	0.00
25	9	485	84.24	463	145.38	482	213.42	10	5.42	10	37.93	28	780.38	8913	0.12	854	0.18
26	17	144	0.05	150	0.03	146	0.06	18	0.00	18	0.00	52	0.28	182	0.00	142	0.00
27	21	616	0.33	675	0.21	650	0.49	22	0.01	22	0.01	179	0.85	773	0.00	616	0.00
28	14	79	0.02	79	0.01	79	0.02	15	0.00	15	0.00	49	0.29	95	0.00	79	0.00
29	21	142772	436.34	187319	307.77	159325	709.89	22	0.39	22	1.43	3337	7.12	244499	1.27	192459	2.32
30	22	1791	1.25	1982	0.80	1883	1.90	23	0.01	23	0.02	393	1.35	2295	0.01	1834	0.01
31	19	11278	25.93	6810	38.66	8297	53.43	2647	0.89	723	6.55	7530	32.97	53911	0.25	16766	0.36
32	24	431	0.17	431	0.10	431	0.25	25	0.00	25	0.00	352	0.74	435	0.00	424	0.00
33	21	1480	0.84	1436	0.30	1391	1.00	446	0.26	22	0.63	947	2.29	2291	0.01	1073	0.01
34	21	223	0.07	223	0.04	223	0.09	22	0.00	22	0.00	158	0.50	227	0.00	216	0.00
35	22	65965	160.36	63186	39.55	68281	199.30	24021	0.83	11113	6.36	7448	8.27	165170	0.63	61548	1.06
36	22	571766	392.49	371834	786.06	458402	1094.61	48350	2.98	2783	14.07	188564	111.99	1669788	9.44	717884	18.27
37	23	1307	1.29	1417	0.95	1363	2.10	24	0.02	24	0.01	277	2.10	1532	0.00	1342	0.01
38	13	301	0.20	372	0.15	326	0.32	14	0.01	14	0.01	33	0.74	562	0.00	357	0.00
39	23	2486	2.49	2942	1.64	2682	3.91	24	0.08	24	0.07	146	1.78	4103	0.01	2597	0.02
40	20			182608	1384.90			38837	1.88	7767	12.86	23371	87.91	1036992	6.74	229210	9.51
41	10	31	0.01	34	0.00	31	0.01	11	0.00	11	0.00	21	0.16	54	0.00	35	0.00
42	30	1855	0.50	1747	0.17	1739	0.59	1117	0.18	31	0.18	1773	1.29	1908	0.01	1636	0.01
43	20	328	0.09	328	0.05	328	0.12	21	0.00	21	0.00	256	0.50	333	0.00	315	0.00
44	19	2990	3.25	3430	2.30	3121	5.24	20	0.05	20	0.05	407	2.18	4142	0.01	3235	0.02
45	20	347	0.16	376	0.11	359	0.25	21	0.01	21	0.00	121	0.74	434	0.00	358	0.00
46	34	60888	51.77	61842	21.14	61563	68.33	36941	0.67	32582	4.05	19865	6.91	80785	0.25	65984	0.63
47	27	4104	5.27	4522	3.93	4284	8.70	28	0.04	28	0.04	515	2.32	5075	0.01	4406	0.02
48	37							129627	2.37	2500	11.08	200559	101.21			19020089	286.02
49	47							2048368	15.84	594399	23.32	2772875	1408.64				
50	23	637	0.39	659	0.26	645	0.60	24	0.02	24	0.02	390	1.40	690	0.00	642	0.00
rovers-ipc5																	
01	10	147	0.01	147	0.01	147	0.02	11	0.03	11	0.03	48	0.07	1104	0.00	283	0.00
02	8	44	0.01	44	0.01	44	0.01	9	0.00	9	0.00	16	0.03	254	0.00	129	0.00
03	11	672	0.11	419	0.05	448	0.10	12	0.11	12	0.12	804	0.16	3543	0.02	757	0.00
04	8	47	0.02	20	0.00	24	0.01	9	0.04	9	0.04	58	0.08	897	0.00	223	0.00
05	22	808084	237.13	410712	123.64	522937	231.28	617267	11.48	375808	18.46	298400	101.65	8559690	126.19	4318309	81.53
07	18			741649	517.18	1682245	1780.27	3280884	51.02	2212903	59.20	1459792	866.93			9618062	199.91
12	19									5187273	166.77						
satellite-ipc4																	
01	9	24	0.00	32	0.00	29	0.00	10	0.00	10	0.00	46	0.06	89	0.00	59	0.00
02	13	86	0.02	337	0.10	241	0.13	14	0.01	14	0.01	646	0.21	1728	0.01	940	0.00
03	11	2249	1.24	656	0.53	728	0.82	12	0.56	12	0.64	1945	0.93	15185	0.17	6822	0.11
04	17	9817	10.65	14860	24.90	11250	26.18	4152	0.99	18	4.43	15890	9.50	345663	4.70	180815	3.37
05	15	279569	1251.83	46453	515.80	61692	877.26	81972	7.26	148667	69.28	267513	565.18				
06	20	1496577	968.24	1572327	1721.87			2769229	74.73	307962	32.52					10751017	371.43
tpp-ipc5																	
01	5	6	0.00	6	0.00	6	0.00	6	0.00	6	0.00	6	0.01	7	0.00	6	0.00
02	8	9	0.00	11	0.00	9	0.00	9	0.00	9	0.00	9	0.01	26	0.00	16	0.00
03	11	12	0.00	27	0.00	16	0.00	12	0.00	12	0.00	12	0.03	116	0.00	83	0.00
04	14	15	0.01	78	0.01	47	0.01	15	0.01	15	0.00	15	0.07	494	0.00	430	0.00
05	19	623	0.52	5110	1.36	1455	1.21	20	0.36	20	0.77	624	0.48	24698	0.12	17398	0.15
06	25							947059	14.22	74798	23.97					9267024	216.69
zenotravel-ipc3																	
01	1	2	0.01	2	0.00	2	0.01	2	0.00	2	0.00	2	0.45	2	0.00	2	0.00
02	6	17	0.02	18	0.02	17	0.02	7	0.00	7	0.00	9	0.46	58	0.00	22	0.00
03	6	28	0.08	18	0.12	12	0.11	7	0.21	7	0.90	40	3.42	5160	0.04	492	0.02
04	8	99	0.15	88	0.26	81	0.30	9	0.20	9	0.89	215	3.44	5256	0.03	665	0.01
05	11	177	0.32	220	0.22	136	0.36	12	0.25	12	1.90	422	7.70	82289	0.63	12466	0.33
06	11	2287	5.51	1144	2.00	504	2.40	12	0.38	12	3.54	1957	11.81	596531	5.90	85931	2.47
07	15	5088	9.63	4234	5.56	4199	10.58	16	0.38	16	3.48	34890	30.36	405626	3.56	115348	2.60
08	11	3268	43.96	1026	8.92	1655	30.06	14354	2.00	12	14.48	83533	292.05			687846	50.76
09	21							2517035	51.18	611457	30.47						
10	22							1322871	34.84	137872	25.44						
11	14			76904	1090.67			310030	11.28	110726	26.65						

Table B.5: Similar to Table B.1 for the PSR, Rovers, Satellite, TPP, and Zenotravel domains.

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
schedule-strips																	
02-0	3	5	0.15	5	0.14	5	0.22	4	511.10	4	1743.32	5	577.39	76	0.02	5	0.09
02-1	2	3	0.16	4	0.11	3	0.18	3	104.98			3	754.26	6	0.02	3	0.07
02-2	2	3	0.32	3	0.17	3	0.40	3	231.99			3	495.56	5	0.02	3	0.07
02-3	3	26	0.50	37	0.76	26	0.61	4	56.51			4	658.90	529	0.03	95	0.45
02-4	3	68	1.34	188	2.24	220	7.20					4	484.62	543	0.03	108	0.44
02-5	2	3	0.33	3	0.14	3	0.38	3	363.11			3	667.32	3	0.03	3	0.07
02-6	2	3	0.14	5	0.12	3	0.17	3	121.84			3	697.42	6	0.02	3	0.06
02-7	2	3	0.30	3	0.13	3	0.34	3	323.77			3	604.06	13	0.02	3	0.07
02-8	2	3	0.32	3	0.14	3	0.38	3	316.53			3	668.79	8	0.02	3	0.07
02-9	3	5	0.15	5	0.14	5	0.22	4	251.46			5	577.16	76	0.03	5	0.09
03-0	4	40	2.72	407	12.16	140	14.55							11915	0.60	1127	8.98
03-1	2	3	0.51	3	0.35	3	0.72							31	0.04	25	0.37
03-2	4	27	1.16	50	1.83	33	2.33	5	191.03					3617	0.23	1228	9.56
03-3	4	15	0.79	91	2.39	15	0.96	5	259.13					3379	0.23	170	1.85
03-4	3	4	1.11	16	2.08	4	1.52					4	1223.90	301	0.06	22	0.27
03-5	4	73	6.13	471	16.71	74	8.32	5	682.30					12217	0.64	1175	12.43
03-6	4	72	1.27	75	1.80	69	1.33	5	121.58					2663	0.19	1542	11.73
03-7	4	28	1.05	50	1.83	28	1.43	5	195.72					12859	0.68	1323	13.47
03-8	4	273	11.53	266	11.46	273	17.48							12616	0.65	1590	11.13
03-9	4	8	0.96	31	1.77	14	2.13	5	235.48					4339	0.27	913	7.69
04-0	5	373	13.91	1498	74.46	167	24.60							312193	26.88	22993	273.38
04-1	6	17559	1373.80	10707	626.54			7	1115.76					552069	49.79		
04-2	5	209	9.88	406	20.85	66	5.30							47696	4.97	9703	131.69
04-3	5	142	10.47	674	33.29	251	29.28	6	267.29					89272	8.74	12941	163.84
04-4	5	921	64.48	450	46.95	574	116.65							62013	6.03	13614	168.07
04-5	6	483	47.25	4544	268.77	850	187.46	7	837.68							107978	1399.99
04-6	6	779	27.09	11610	361.74	1834	102.68	7	459.19							107115	1001.40
04-7	5	99	18.48	424	38.04	163	40.04	6	936.68					61327	5.97	8683	103.50
04-8	5	102	16.01	573	31.87	111	23.35	6	711.65					340467	29.56	15122	181.98
04-9	4	1043	80.06	996	76.64	1050	143.48	5	316.22					41673	4.27	5480	83.69
05-0	5	163	41.61	483	63.23	167	62.53							143350	22.71	43336	751.35
05-1	6	2701	213.92			1257	286.28										
05-3	7					13622	1693.68										
05-4	6	989	100.02	3433	229.05	582	100.05										
05-5	6	198	21.67	9550	767.94	347	68.64									120602	989.42
05-6	7	6033	743.61			10325	1508.56										
05-7	6	944	131.19	17562	1446.20	2107	379.70										
05-8	7	1190	172.59			2709	730.54										
05-9	6	1537	140.49	15829	1248.19	2717	547.56										
06-2	6	888	243.14			1709	730.36										
06-4	8	11535	1776.87														
07-0	7	2489	786.76														
07-9	8	6829	1559.86														

Table B.6: Similar to Table B.1 for the (non-IPC) SCHEDULE-STRIPS domain.

B.2 Databased Fork-Decompositon

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
airport-ipc4																	
01	8	10	0.01	9	0.00	9	0.00	9	0.00	9	0.00	9	0.72	11	0.00	9	0.00
02	9	12	0.01	15	0.00	15	0.01	10	0.00	10	0.00	10	1.23	13	0.00	10	0.00
03	17	86	0.02	133	0.01	93	0.02	18	0.04	18	0.03	29	5.10	164	0.00	57	0.00
04	20	22	0.01	21	0.00	21	0.01	21	0.02	21	0.01	21	1.32	23	0.00	21	0.00
05	21	23	0.08	30	0.02	27	0.09	22	0.01	22	0.01	22	46.54	27	0.00	22	0.00
06	41	513	0.16	639	0.06	567	0.19	42	0.16	42	0.17	42	123.13	738	0.01	418	0.02
07	41	514	0.15	632	0.05	550	0.19	42	0.17	42	0.17	42	117.56	742	0.01	405	0.02
08	62	12733	1.89	21544	1.36	14398	4.02	24372	25.42	9623	1549.13	203	602.09	27032	0.28	9687	0.90
09	71	88670	16.58	136717	9.60	90412	38.78	152408	64.92	89525	466.14	12956	993.07	175717	2.47	56484	7.62
10	18	19	0.01	19	0.01	19	0.01	19	0.02	19	0.01	19	2.45	21	0.00	19	0.00
11	21	23	0.10	30	0.03	27	0.12	22	0.02	22	0.01	22	65.36	27	0.00	22	0.01
12	39	475	0.20	728	0.07	568	0.25	40	0.21	40	0.21	40	169.02	873	0.01	392	0.03
13	37	434	0.20	663	0.07	479	0.24	38	0.20	38	0.21	38	134.87	822	0.01	342	0.03
14	60	12040	2.90	25110	1.86	15948	4.64	30637	51.23	8968	238.16	62	714.76	35384	0.39	9196	1.11
15	58	11477	2.74	23317	1.71	14557	4.25	28798	46.20	8931	267.81	59	647.05	33798	0.38	8200	1.01
16	79	267277	77.39	824491	97.12	353592	114.58	1031524	200.95	305340	1077.90			1247467	19.72	221993	49.03
17	88	2460667	708.82			2678689	1235.79									1043661	310.89
19	90	1354353	592.53	3400142	492.06	1462739	660.17									831632	253.21
21	101	5156	48.29	11259	3.72	4773	51.13	7326	372.92	102	10.28			18809	0.42	3184	1.12
22	148	606648	1110.09	1063668	318.90	477836	1082.91	1119943	762.02							159967	105.29
36	109	9504	129.73	34986	14.41	9436	140.75	34365	853.70					63061	1.44		
37	142	37873	820.33														
blocks-ipc2																	
04-0	6	15	0.00	46	0.00	17	0.00	7	0.03	7	0.03	7	0.36	93	0.00	25	0.00
04-1	10	14	0.00	31	0.00	15	0.00	11	0.04	11	0.03	11	0.39	66	0.00	23	0.00
04-2	6	7	0.00	26	0.00	10	0.00	7	0.04	7	0.03	7	0.38	63	0.00	18	0.00
05-0	12	32	0.00	302	0.01	113	0.00	13	0.30	13	0.96	13	1.32	467	0.00	145	0.00
05-1	10	37	0.00	280	0.00	98	0.00	11	0.29	11	0.96	11	1.36	567	0.00	135	0.00
05-2	16	152	0.00	596	0.00	348	0.01	17	0.29	17	0.95	17	1.49	792	0.00	297	0.00
06-0	12	33	0.00	766	0.01	207	0.01	13	0.95	13	8.56	13	4.10	1826	0.00	276	0.00
06-1	10	41	0.00	2395	0.03	578	0.02	11	0.90	11	8.34	11	4.17	4887	0.01	755	0.01
06-2	20	855	0.01	5444	0.05	3352	0.06	733	0.87	85	8.84	31	4.29	6385	0.02	2556	0.03
07-0	20	278	0.01	20183	0.28	4022	0.12	577	1.93	144	23.32	22	11.47	37157	0.14	5943	0.11
07-1	22	6910	0.10	59207	0.60	38539	0.67	10071	1.70	1835	21.05	174	11.25	63376	0.21	33194	0.46
07-2	20	1458	0.02	46009	0.52	18854	0.39	1855	1.59	782	20.37	90	10.99	55218	0.19	18293	0.29
08-0	18	1533	0.03	344157	5.46	69830	2.09	5557	3.67	678	36.80	25	26.00	519107	2.28	94671	2.07
08-1	20	10040	0.17	517514	7.22	191352	4.91	45711	3.88	11827	33.49	151	26.57	636498	2.60	199901	3.85
08-2	16	479	0.02	237140	4.08	32567	1.09	277	3.63	54	32.53	17	25.85	433144	1.93	52717	1.30
09-0	30	134185	3.10	7405904	117.14	4346535	118.23	1233374	16.00	971409	77.74	464	56.76	7984649	36.76	3840589	85.00
09-1	28	3435	0.09	4145371	77.54	917197	33.32	95068	7.35	58873	63.15	82	56.98	5914572	29.73	1200345	32.06
09-2	26	6379	0.17	4145278	78.21	923365	33.79	161719	13.54	20050	82.45	81	57.02	5963160	30.02	1211463	32.15
10-0	34	1524599	36.52									1800	114.26				
10-1	32	610206	15.79							12063665	228.76	1835	115.19				
10-2	34	1516087	37.71									3685	116.75				
11-0	32									7046739	141.44	2678	213.32				
11-1	30											1510	203.79				
11-2	34											3984	213.97				
12-0	34											1184	370.06				
12-1	34											614	382.34				
13-0	42											83996	860.45				
13-1	44											163438	1104.27				
14-0	38											2779	1063.02				
14-1	36											7154	1087.40				
driverlog-ipc3																	
01	7	49	0.00	37	0.00	37	0.00	8	0.04	8	0.03	44	0.47	182	0.00	20	0.00
02	19	15713	0.42	18452	0.27	15794	0.55	20	0.13	20	0.26	15998	4.55	68927	0.36	54283	0.52
03	12	164	0.00	190	0.00	163	0.01	13	0.16	13	0.25	863	1.25	16031	0.09	2498	0.03
04	16	6161	0.42	10778	0.30	7665	0.62	17	0.49	17	2.41	22933	12.20	999991	8.12	393673	6.56
05	18	13640	1.01	11400	0.36	10984	1.07	2614	0.60	19	4.58	24877	18.77	6290803	61.57	1724611	34.73
06	11	608	0.09	795	0.06	492	0.11	291	1.35	12	9.72	3804	10.08	681757	7.64	54451	1.71
07	13	864	0.14	1730	0.11	1006	0.21	14	1.42	14	15.35	25801	41.34	6349767	81.53	493480	17.31
08	22	669994	75.74	1181268	61.32	694996	104.59	287823	7.34	2952	20.31						
09	22	150255	14.72	198651	11.44	164109	23.06	15504	1.70	23	10.43						
10	17	4304	0.44	16099	1.21	4037	0.69	18	1.64	18	18.54	18234	68.22				
11	19	43395	4.99	41445	2.22	39069	5.90	34137	1.99	10790	17.01	559623	1193.00			6141130	330.22
13	26	1303099	325.71	1014865	144.64	1098694	422.20	1298884	19.52	870875	35.33						

Table B.7: Runtimes of cost-optimal heuristic-search planners on the AIRPORT, BLOCKSWORLD, and DRIVERLOG domains. The description of the planners is given in Section 4.4.2; here the fork-decomposition heuristics are via databased implicit abstractions. Column *task* denotes problem instance, column h^* denotes optimal solution length. Other columns capture the run *time* and number of expanded *nodes*.

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
depots-ipc3																	
01	10	114	0.01	279	0.01	161	0.02	11	0.00	11	0.00	45	0.77	329	0.00	136	0.00
02	15	1134	0.08	9344	0.31	2638	0.22	738	3.24	16	1.14	898	11.56	15404	0.11	3771	0.17
03	27	134428	8.59	2520703	159.84	581726	66.43	348288	20.69	239313	222.35	103089	247.13	2930398	27.20	1204646	97.62
04	30	1254545	101.18			5835295	923.87	1284048	52.05	1273762	529.34						
07	21	109765	9.17	4271196	336.59	487961	76.02	211820	37.54			41328	324.19	6501100	71.58	1331701	166.76
10	24	2964635	283.55			6081478	1187.66	3241083	157.52								
13	25	1003709	152.30			8161872	1559.21	1427824	116.06								
freecell-ipc3																	
01	8	234	0.10	974	0.15	274	0.17	87	3.12	9	38.74	9	13.01	3437	0.03	1043	0.15
02	14	30960	1.95	75150	5.53	37131	4.79	31487	40.40			466	70.29	130883	1.46	41864	10.77
03	18	197647	14.41	533995	78.27	240161	51.24	95805	140.96			1589	169.39	944843	11.45	210503	75.62
04	26	997836	60.67	1921470	232.95	1218329	213.02	943074	86.78			15848	341.02	3021326	38.80	600525	247.70
05	30	6510089	448.22					5950977	243.74			40642	916.44			1408035	1062.25
grid-ipc1																	
01	14	571	0.60	1117	0.34	472	0.78	660	8.63	467	121.10			6446	0.08	190	0.10
02	26	3330274	1078.55					3392724	50.35	3244132	241.94					664016	231.26
gripper-ipc1																	
01	11	214	0.00	240	0.00	214	0.00	12	0.00	12	0.00	33	0.11	236	0.00	208	0.00
02	17	1768	0.02	1832	0.01	1803	0.03	18	0.11	18	0.08	680	0.37	1826	0.01	1760	0.01
03	23	11626	0.19	11736	0.08	11689	0.22	11514	0.47	2094	1.75	7370	1.52	11736	0.04	11616	0.08
04	29	68380	1.46	68558	0.51	68479	1.63	68380	1.24	68190	8.05	55568	10.29	68558	0.27	68368	0.56
05	35	376510	10.07	376784	3.20	376653	11.11	376510	3.52	376510	19.46	344386	79.96	376772	1.59	376496	3.51
06	41	1982032	70.91	1982408	19.08	1982227	77.81	1982032	13.42	1982032	42.16	1911592	577.49	1982394	9.59	1982016	21.57
07	47	10091986	438.41	10092464	105.67	10092241	478.67	10091986	61.66	10091986	106.84			10092464	51.10	10091968	119.64
logistics-ipc1																	
01	26	77763	7.14	1469610	95.49	830292	98.59	1918881	41.03	949586	34.82	211955	1700.26				
05	22	3293	0.46	850312	42.43	173477	18.19	768161	18.69	609393	35.27						
31	13	436	0.03	1981	0.07	1284	0.09	494	0.42	14	2.11	481	6.58	155645	1.66	32282	0.57
32	20	392	0.01	2704	0.07	962	0.05	21	0.16	21	0.72	9598	7.08	245325	2.07	81156	1.00
33	27	312180	27.19			3617185	427.52			529338	32.55						
35	30	477883	183.08														
logistics-ipc2																	
04-0	20	21	0.00	193	0.00	65	0.00	21	0.03	21	0.05	21	0.34	11246	0.05	4884	0.03
04-1	19	20	0.00	570	0.01	293	0.00	20	0.03	20	0.04	20	0.37	9249	0.04	4185	0.03
04-2	15	16	0.00	117	0.00	79	0.00	16	0.04	16	0.05	16	0.36	4955	0.02	1205	0.01
05-0	27	28	0.00	2550	0.05	1171	0.03	28	0.10	28	0.38	28	0.58	109525	0.64	74694	0.59
05-1	17	18	0.00	675	0.01	427	0.01	18	0.10	18	0.38	18	0.72	22307	0.13	6199	0.05
05-2	8	9	0.00	24	0.00	13	0.00	9	0.09	9	0.38	9	0.78	1031	0.00	280	0.00
06-0	25	26	0.00	4249	0.09	2461	0.07	26	0.18	26	1.23	26	1.03	490207	3.40	202229	1.92
06-1	14	15	0.00	181	0.00	99	0.00	15	0.18	15	1.26	15	1.16	24881	0.16	3604	0.03
06-2	25	26	0.00	2752	0.06	1394	0.04	26	0.19	26	1.26	26	1.03	476661	3.32	200012	1.98
06-9	24	25	0.00	2395	0.04	1428	0.04	25	0.18	25	1.22	25	1.02	422557	2.95	133521	1.29
07-0	36	37	0.00	251287	7.52	98053	4.59	525	0.65	37	4.87	24317	35.46				
07-1	44	1689	0.07	3532213	99.33	1705009	72.35	666324	8.83	49	4.94	362179	453.06				
08-0	31	32	0.00	82476	2.69	35805	1.78	1042	0.96	32	6.90	14890	33.50				
08-1	44	45	0.01	1183608	45.72	462244	25.36	16708	1.15	45	7.21	114155	198.84				
09-0	36	37	0.00	351538	13.75	167038	9.76	20950	1.56	37	9.46	32017	83.16				
09-1	30	31	0.00	59336	2.48	25359	1.73	31	1.27	31	9.43	6720	26.48				
10-0	45	46	0.01							668834	29.73						
10-1	42	43	0.01							1457130	43.00						
11-0	48	697	0.09							701106	37.42						
11-1	60	21959	2.22														
12-0	42	43	0.02							775996	43.56						
12-1	68	106534	11.64							2222340	87.47						
mprime-ipc1																	
01	5	196	0.02	10	0.01	24	0.01	6	2.00	6	20.45	108	49.59	3636	0.07	68	0.04
02	7	11604	2.72	44045	80.68	2565	4.20	3317	88.58							12606	36.65
03	4	427	0.27	7	0.08	11	0.16	36	33.64	5	463.85			9868	0.67	5	0.07
04	8	3836	0.22	1775	0.10	1093	0.09	9	6.09	9	82.71	19076	781.74	599590	23.58	200	0.24
05	11	1745027	195.08			604756	592.60	1705009	127.53							1488157	1638.78
07	5	3314	0.25	47	0.03	346	0.08	1667	46.72					18744	0.56	11	0.04
08	6	485381	491.53	1376780	1426.21			1469752	403.45							7650	84.33
09	8	19838	2.92	100188	74.85	5227	6.31	21993	36.25					2197646	71.69	19023	30.26
11	7	9	0.02	219	0.03	8	0.03	8	4.69	8	62.68	22	394.26	73260	2.21	915	0.54
12	6	16320	1.89	8118	0.73	5243	1.13	34763	11.45	42055	143.27	25665	724.12	108652	3.50	1520	1.78
15	6															1039	178.55
16	6	252	0.76	51590	135.00	448	2.76	473	81.42					425144	32.17	7962	35.65
17	4	2746	10.47	453	18.78	451	21.40							172736	42.48	5	1.06
19	6	727401	521.78	95361	485.79			123039	313.25							36013	533.75
21	6	174221	55.09	34022	47.43	169400	392.30							1503293	103.23	15250	101.75
25	4	75	0.01	30	0.01	29	0.01	5	0.48	5	2.75	85	8.71	383	0.00	6	0.00
26	6	77622	24.69	147854	48.25	68239	106.35	172432	46.33	189154	454.69			819590	61.01	440	2.69
27	5	54	0.16	1772	1.50	9	0.18	6	11.59	6	154.43			84079	3.50	831	2.08
28	7	8	0.01	403	0.02	37	0.02	8	1.88	8	22.55	128	146.80	17333	0.25	211	0.06
29	4	182	0.12	56	0.08	32	0.11	5	14.92	5	201.40			3187	0.17	7	0.10
31	4	248	0.51	46	0.68	19	1.00	419	99.87					3584	0.19	11	0.17
32	7	31759	1.73	12436	1.46	11839	1.93	19429	21.61	7269	292.37	11073	1701.00	115479	2.75	3096	1.74
34	4	234	0.26	46	0.16	23	0.28	450	151.69					3618	0.19	11	0.18
35	5	392	0.07	290	0.06	84	0.08	359	3.63	6	43.43	706	96.55	2476	0.05	44	0.03

Table B.8: Similar to Table B.7 for the Depots, Freecell, Grid, Gripper, Logistics-ipc1, Logistics-ipc2, and Mprime domains.

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
miconic-strips-ipc2																	
01-0	4	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.01	5	0.00	5	0.00
01-1	3	5	0.00	5	0.00	5	0.00	4	0.00	4	0.00	4	0.00	5	0.00	4	0.00
01-2	4	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.01	5	0.00	5	0.00
01-3	4	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00
01-4	4	5	0.00	5	0.00	5	0.00	5	0.00	5	0.00	5	0.01	5	0.00	5	0.00
02-0	7	19	0.00	22	0.00	19	0.00	8	0.00	8	0.00	26	0.01	30	0.00	20	0.00
02-1	7	21	0.00	23	0.00	21	0.00	8	0.00	8	0.00	26	0.01	30	0.00	22	0.00
02-2	7	21	0.00	23	0.00	21	0.00	8	0.00	8	0.00	27	0.00	30	0.00	22	0.00
02-3	7	24	0.00	24	0.00	24	0.00	8	0.00	8	0.00	20	0.01	26	0.00	17	0.00
02-4	7	19	0.00	22	0.00	19	0.00	8	0.00	8	0.00	23	0.01	31	0.00	20	0.00
03-0	10	86	0.00	129	0.00	98	0.00	11	0.00	11	0.00	100	0.03	193	0.00	105	0.00
03-1	11	120	0.00	168	0.00	147	0.00	12	0.00	12	0.00	140	0.02	218	0.00	150	0.00
03-2	10	137	0.00	143	0.00	137	0.00	11	0.00	11	0.00	122	0.02	164	0.00	92	0.00
03-3	10	96	0.00	153	0.00	117	0.00	11	0.00	11	0.00	131	0.02	197	0.00	130	0.00
03-4	10	103	0.00	149	0.00	115	0.00	11	0.00	11	0.00	114	0.02	190	0.00	114	0.00
04-0	14	524	0.00	843	0.00	686	0.01	15	0.01	15	0.01	669	0.10	1182	0.00	866	0.00
04-1	13	505	0.00	817	0.00	663	0.01	14	0.01	14	0.01	634	0.11	1176	0.00	860	0.00
04-2	15	685	0.00	942	0.00	802	0.01	16	0.01	16	0.01	822	0.12	1277	0.00	969	0.00
04-3	15	681	0.00	942	0.00	798	0.01	16	0.01	16	0.01	820	0.12	1319	0.00	970	0.00
04-4	15	685	0.00	942	0.00	802	0.01	16	0.01	16	0.01	821	0.12	1334	0.00	969	0.00
05-0	17	2468	0.03	4009	0.03	3307	0.05	18	0.06	18	0.05	2829	0.44	6350	0.03	4387	0.03
05-1	17	2807	0.04	4345	0.03	3677	0.06	18	0.06	18	0.05	3260	0.49	6602	0.03	4664	0.03
05-2	15	1596	0.02	2981	0.02	2275	0.04	16	0.06	16	0.05	1594	0.32	5565	0.03	3524	0.03
05-3	17	2256	0.03	3799	0.03	3104	0.05	18	0.06	18	0.05	2568	0.42	5944	0.03	4140	0.03
05-4	18	3210	0.04	4732	0.03	4267	0.06	19	0.06	19	0.05	3953	0.55	6949	0.04	5268	0.04
06-0	19	9379	0.18	17665	0.15	13531	0.26	20	0.18	20	0.32	9312	1.76	30786	0.20	21194	0.20
06-1	19	9106	0.17	18134	0.15	14052	0.27	20	0.18	20	0.32	10252	1.96	30093	0.20	21255	0.20
06-2	20	10900	0.20	19084	0.16	15111	0.28	21	0.18	21	0.32	11247	2.11	32390	0.21	21694	0.21
06-3	20	12127	0.23	21708	0.18	17807	0.33	21	0.17	21	0.32	14216	2.56	32574	0.21	24552	0.23
06-4	21	13784	0.24	23255	0.19	19536	0.35	22	0.17	22	0.32	16880	3.04	33793	0.22	26167	0.24
07-0	23	53662	1.19	96092	0.97	79449	1.76	24	0.32	24	1.75	56686	14.31	155466	1.22	116685	1.32
07-1	24	56328	1.24	99109	0.96	83677	1.83	7001	0.38	25	1.75	63035	16.33	164470	1.29	118494	1.33
07-2	22	48141	1.10	96139	0.94	78471	1.77	1646	0.33	23	1.71	55751	13.98	161342	1.27	119688	1.36
07-3	22	46867	1.08	93117	0.92	75424	1.69	1861	0.33	23	1.74	53121	13.27	155176	1.23	114649	1.30
07-4	25	84250	1.70	126595	1.22	111984	2.36	23159	0.52	26	1.71	96327	24.76	168219	1.33	140128	1.58
08-0	27	272580	7.05	485051	5.51	408114	10.53	41629	0.91	28	4.18	290649	104.18	755255	7.16	594032	7.95
08-1	27	284415	7.56	527216	6.01	446837	11.58	42679	0.90	28	4.25	339177	123.10	794365	7.56	636587	8.66
08-2	26	207931	5.60	414294	4.79	330993	8.90	37744	0.86	27	4.25	204614	73.39	731622	6.92	534711	7.37
08-3	28	369479	9.25	598031	6.74	527216	13.30	140453	1.94	29	4.21	435617	160.49	833421	7.97	690267	9.29
08-4	27	297516	7.74	507910	5.79	431432	11.04	62933	1.16	28	4.12	315339	111.84	771608	7.33	613253	8.43
09-0	31	1461729	43.82	2491975	32.67	2138656	63.58	684737	9.07	126918	8.89	1555286	794.93	3685552	41.04	3006991	49.12
09-1	30	1207894	37.47	2335166	30.76	1952916	59.39	406041	5.61	100937	8.73	1344815	683.05	3649801	40.32	2893803	47.54
09-2	30	1294691	40.03	2340411	30.97	1972234	59.25	442547	6.06	82946	8.63	1357681	692.11	3576134	39.61	2895182	47.26
09-3	32	1840936	52.68	2889342	38.12	2571844	74.47	765455	10.00	277302	11.14	2083168	1051.95	3796035	42.13	3304570	53.29
09-4	28	1252484	40.34	2352633	31.35	1944297	59.37	317692	4.65	29	7.03	1231554	605.01	3589382	39.29	2956995	48.84
10-0	33	5716041	202.37	10316603	153.80	8774563	300.08	2436164	35.24	863244	23.76			15804498	200.90	13267920	250.58
10-1	32	5601282	201.43	10789013	162.69	9144153	315.23	2340169	34.09	335745	15.68			16472633	208.39	13720664	256.89
10-2	32	4153191	155.86	9148616	138.69	7466572	265.86	1735477	25.29	486286	17.72			15867374	201.01	12497087	236.89
10-3	34	6108094	214.68	10960203	167.10	9400386	320.13	3952148	55.86	940556	24.24			16309701	208.42	13801989	262.53
10-4	33	5920127	211.40	11075136	170.82	9448049	322.74	2715866	39.44	625559	19.91			16472551	209.13	13925654	262.57
11-0	37							11473359	183.60	4724980	93.56						
11-1	34	15349953	668.77					7535468	124.80	1934943	47.91						
11-2	38							14645785	233.68	6330198	120.71						
11-3	38									5809711	110.10						
11-4	35							5853546	95.56	1082086	32.22						
mystery-ipc1																	
01	5	7	0.00	6	0.00	6	0.00	6	0.20	6	1.79	10	5.38	30	0.00	8	0.00
02	7	2404	0.50	8012	11.19	722	1.01	1672	82.70					770852	21.85	2368	4.47
03	4	73	0.08	7	0.04	11	0.10	5	16.46	5	193.75	65	811.87	507	0.02	5	0.03
04	∞	0	0.00			0	0.00										
07	∞	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
09	8	3049	0.37	10764	5.66	1215	1.01	3165	29.34			3868	670.08	138289	2.18	1458	1.44
11	7	9	0.01	33	0.01	8	0.01	8	1.51	8	16.59	34	41.20	426	0.00	19	0.00
12	∞	2102777	33.84	2093419	55.58	2093419	76.80	2102777	14.61	2102729	27.84			2102777	15.09	1177842	21.87
15	6	28271	20.21	21572	41.22	5079	44.42							279973	13.21	135	2.62
16	∞	0	0.15			0	0.27										
17	4	354	1.32	85	2.74	83	3.59	198	445.85					5400	0.41	5	0.35
18	∞	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00	0	0.00
19	6	21717	4.87	4968	5.26	16276	29.28	12478	96.38					133871	3.65	1516	5.44
20	7	89887	46.32	84572	153.53	53114	173.34	285069	59.22	547246	578.39			686125	23.28	718	3.76
24	∞	0	0.13			0	0.30										
25	4	9	0.00	10	0.00	9	0.01	5	0.10	5	0.10	14	1.22	31	0.00	6	0.00
26	6	1807	0.27	1835	0.30	1344	0.69	2526	5.94	346	70.78	3107	291.36	8455	0.10	37	0.05
27	5	14	0.05	159	0.09	6	0.07	6	4.80	6	80.48	7	243.78	2174	0.03	73	0.04
28	7	8	0.00	47	0.00	15	0.00	8	0.63	8	6.77	31	16.67	843	0.00	32	0.00
29	4	31	0.04	14	0.03	10	0.06	5	8.94	5	107.10	27	536.30	153	0.01	7	0.02
30	9	23175	5.16	76480	169.86	7232	13.30	42112	28.07	44893	357.07			1977063	38.26	26686	28.27

Table B.9: Similar to Table B.7 for the MICONIC and MYSTERY domains.

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
openstacks-ipc5																	
01	23	2264	0.02	3895	0.03	3070	0.05	24	0.05	24	0.06	2000	1.02	4822	0.01	4016	0.03
02	23	2617	0.03	4485	0.04	3561	0.05	24	0.06	24	0.06	2378	1.07	5501	0.02	4594	0.04
03	23	2264	0.02	3895	0.03	3070	0.05	24	0.06	24	0.06	2000	1.02	4822	0.01	4016	0.03
04	23	2264	0.02	3895	0.03	3070	0.05	24	0.06	24	0.06	2000	1.02	4822	0.02	4016	0.03
05	23	2264	0.02	3895	0.03	3070	0.05	24	0.06	24	0.05	2000	1.02	4822	0.01	4016	0.03
06	45	366768	7.52	779710	18.93	587482	22.20	621008	4.85	279614	7.86	379735	217.37	882874	4.91	822514	18.71
07	46	410728	8.23	760668	18.33	606782	22.53	594758	4.69	264535	7.34	405564	226.32	836647	4.62	787163	17.81
pathways-ipc5																	
01	6	1624	0.00	1299	0.00	1299	0.00	7	1.14	7	0.79	1405	0.28	1624	0.00	36	0.00
02	12	2755	0.02	2307	0.01	2437	0.02	1946	2.56	13	42.11	990	0.29	2984	0.02	348	0.01
03	18	44928	0.62	20416	0.25	29106	0.43	21671	6.43	14901	129.23	14772	6.99	87189	1.06	4346	0.16
04	17	126950	2.66	33788	0.59	58738	1.31			98484	288.39	34206	27.00	456143	8.22	104068	2.61
pipesworld-notankage-ipc4																	
01	5	121	0.02	109	0.01	121	0.02	6	0.04	6	0.04	6	2.79	121	0.00	13	0.00
02	12	1413	0.06	1542	0.02	1413	0.08	169	0.30	13	0.17	435	3.07	1808	0.01	792	0.02
03	8	1742	0.14	3001	0.07	1742	0.18	9	1.15	9	0.69	128	3.84	3293	0.02	262	0.02
04	11	7007	0.45	8911	0.22	7007	0.59	651	1.95	12	7.05	812	8.84	16088	0.11	2925	0.13
05	8	4093	0.49	6805	0.26	4093	0.65	77	5.63	9	21.15	155	16.53	11128	0.12	1121	0.15
06	10	12401	1.44	27377	1.34	12401	2.03	1299	5.26	61	39.31	1151	23.41	49905	0.48	7102	0.72
07	8	4370	0.97	9168	0.77	4370	1.34	233	19.78	9	59.70	185	29.88	46502	0.57	2631	0.48
08	10	18851	3.84	56189	6.21	20584	6.42	561	12.42	497	94.69	1673	48.84	273585	3.39	22874	3.58
09	13	1092472	160.71	2419903	151.99	1092472	219.75	104875	25.48			10478	74.26	5513309	80.62	321861	68.99
10	18							2982520	66.89			689832	1439.64			11121245	1579.77
11	20	313952	27.68	472950	29.55	313952	43.90	90598	9.20	52159	43.24	108503	625.52	710123	3.86	107061	14.51
12	24	684234	75.72	1319980	133.58	686186	145.41	594661	12.41	416184	109.43	433296	1117.57	2467804	13.83	464982	56.82
13	16	39998	6.02	117475	18.08	40226	12.69	12835	34.28			24224	1019.65	481045	3.14	33417	6.38
14	30							13255718	119.54								
15	26	1594863	254.43	2588849	192.90	1594863	353.40	648132	65.43					4921698	34.90	555619	105.49
17	22	5437393	1588.68					3200672	90.07								
19	24							8767431	150.88								
21	14	23833	4.02	49035	7.76	23833	7.87	3992	18.13	948	159.63			157782	1.31	8966	2.42
23	18	2285790	568.93	7047138	871.03	2282678	843.28	296506	49.11	104750	256.13					481859	229.00
24	24							7315150	142.82								
41	12	502308	370.68			502308	1092.50									114257	250.18
pipesworld-tankage-ipc4																	
01	5	77	0.02	126	0.01	105	0.02	6	3.54	6	0.13	6	3.88	128	0.00	13	0.01
02	12	960	0.05	1005	0.02	960	0.06	110	3.04	13	0.20	179	6.04	1012	0.01	659	0.02
03	8	20803	1.89	52139	2.46	20803	2.82	244	22.64	9	36.89	818	24.47	52983	0.77	1802	1.33
04	11	110284	8.06	157722	9.60	110284	14.05	3892	16.68	12	155.03	8116	64.68	221429	3.06	41540	14.49
05	8	6531	0.86	13148	1.03	6531	1.32	376	15.46	9	120.06	313	59.99	12764	0.21	2834	1.61
06	10	20171	2.41	43583	4.32	20171	4.41	1794	328.18	11	201.44	3102	97.31	58487	0.87	15746	6.61
07	8	202706	73.83	2643752	1379.11	202706	208.81					2695	339.76	5404036	198.08	104531	420.47
08	11							96043	191.77								
11	22	2345399	296.87	2629204	662.94	2365735	838.85	660104	28.60	660102	162.93			4116344	30.67	752867	334.42
13	16							188517	122.11								
15	30	9652091	1721.67					2546587	141.12								
17	44							12850247	352.46								
21	14	839847	250.39					13241	69.80					4423951	65.44	126845	222.23
31	39	1501847	240.38	1568963	661.88	1504072	850.16	1357801	124.64					1726598	13.56	919764	381.66
rovers-ipc5																	
01	10	147	0.00	147	0.00	147	0.00	11	0.03	11	0.03	48	0.07	1104	0.00	283	0.00
02	8	44	0.00	44	0.00	44	0.00	9	0.00	9	0.00	16	0.03	254	0.00	129	0.00
03	11	672	0.01	419	0.00	448	0.01	12	0.11	12	0.12	804	0.16	3543	0.02	757	0.00
04	8	47	0.00	20	0.00	24	0.00	9	0.04	9	0.04	58	0.08	897	0.00	223	0.00
05	22	808084	22.61	410712	9.23	522937	18.29	617267	11.48	375808	18.46	298400	101.65	8559690	126.19	4318309	81.53
07	18	4546797	191.34	741649	21.01	1682245	102.77	3280884	51.02	2212903	59.20	1459792	866.93			9618062	199.91
12	19			1529551	76.46					5187273	166.77						
satellite-ipc4																	
01	9	24	0.00	32	0.00	29	0.00	10	0.00	10	0.00	46	0.06	89	0.00	59	0.00
02	13	86	0.00	337	0.00	241	0.01	14	0.01	14	0.01	646	0.21	1728	0.01	940	0.00
03	11	2249	0.08	656	0.01	728	0.04	12	0.56	12	0.64	1945	0.93	15185	0.17	6822	0.11
04	17	9817	0.57	14860	0.38	11250	0.76	4152	0.99	18	4.43	15890	9.50	345663	4.70	180815	3.37
05	15	279569	49.47	46453	4.92	61692	18.85	81972	7.26	148667	69.28	267513	565.18				
06	20	1496577	92.22	1572327	51.68	1518261	105.65	2769229	74.73	307962	32.52					10751017	371.43

Table B.10: Similar to Table B.7 for the Openstacks, Pathways, Pipesworld-NoTankage, Pipesworld-Tankage, Rovers, and Satellite domains.

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
psr-small-ipc4																	
01	8	10	0.00	10	0.00	10	0.00	9	0.00	9	0.00	9	0.01	11	0.00	9	0.00
02	11	52	0.00	55	0.00	52	0.00	12	0.00	12	0.00	20	0.08	71	0.00	47	0.00
03	11	31	0.00	31	0.00	31	0.00	12	0.00	12	0.00	20	0.04	33	0.00	28	0.00
04	10	66	0.00	91	0.00	73	0.00	11	0.00	11	0.00	12	0.34	332	0.00	102	0.00
05	11	75	0.00	79	0.00	75	0.00	12	0.00	12	0.00	23	0.11	154	0.00	69	0.00
06	8	10	0.00	10	0.00	10	0.00	9	0.00	9	0.00	9	0.01	11	0.00	9	0.00
07	11	61	0.00	61	0.00	61	0.00	12	0.00	12	0.00	26	0.09	122	0.00	62	0.00
08	8	24	0.00	29	0.00	25	0.00	9	0.00	9	0.00	9	0.12	128	0.00	52	0.00
09	8	18	0.00	19	0.00	18	0.00	9	0.00	9	0.00	9	0.06	49	0.00	20	0.00
10	7	131	0.01	183	0.00	155	0.01	8	0.04	8	0.04	18	1.04	1358	0.00	376	0.01
11	19	149	0.00	149	0.00	149	0.00	20	0.00	20	0.00	96	0.19	153	0.00	142	0.00
12	16	120	0.00	123	0.00	120	0.00	17	0.00	17	0.00	40	0.17	153	0.00	113	0.00
13	15	90	0.00	90	0.00	90	0.00	16	0.00	16	0.00	59	0.16	95	0.00	86	0.00
14	9	19	0.00	19	0.00	19	0.00	10	0.00	10	0.00	13	0.06	27	0.00	18	0.00
15	10	1200	0.08	708	0.03	769	0.09	11	0.46	11	2.58	356	18.99	3562	0.02	324	0.02
16	25	2328	0.02	2158	0.01	2176	0.03	975	0.11	26	0.12	2287	1.34	2742	0.01	1876	0.01
17	9	15	0.00	15	0.00	15	0.00	10	0.00	10	0.00	13	0.03	16	0.00	14	0.00
18	12	85	0.00	90	0.00	85	0.00	13	0.00	13	0.00	29	0.21	158	0.00	91	0.00
19	25	8025	0.11	7856	0.05	7876	0.12	2910	0.27	26	0.77	6338	4.46	9009	0.04	6925	0.08
20	17	80	0.00	80	0.00	80	0.00	18	0.00	18	0.00	52	0.18	84	0.00	75	0.00
21	10	28	0.00	28	0.00	28	0.00	11	0.00	11	0.00	21	0.12	42	0.00	31	0.00
22	33	163299	4.17	176058	1.56	168685	5.01	34	0.28	34	0.87	22315	8.16	189516	0.67	177138	1.43
23	12	77	0.00	93	0.00	77	0.00	13	0.00	13	0.01	30	0.43	200	0.00	116	0.00
24	10	28	0.00	28	0.00	28	0.00	11	0.00	11	0.00	21	0.12	42	0.00	31	0.00
25	9	485	3.06	463	0.58	482	3.28	10	5.42	10	37.93	28	780.38	8913	0.12	854	0.18
26	17	144	0.00	150	0.00	146	0.00	18	0.00	18	0.00	52	0.28	182	0.00	142	0.00
27	21	616	0.01	675	0.00	650	0.01	22	0.01	22	0.01	179	0.85	773	0.00	616	0.00
28	14	79	0.00	79	0.00	79	0.00	15	0.00	15	0.00	49	0.29	95	0.00	79	0.00
29	21	142772	4.55	187319	2.12	159325	5.80	22	0.39	22	1.43	3337	7.12	244499	1.27	192459	2.32
30	22	1791	0.03	1982	0.01	1883	0.04	23	0.01	23	0.02	393	1.35	2295	0.01	1834	0.01
31	19	11278	0.25	6810	0.08	8297	0.24	2647	0.89	723	6.55	7530	32.97	53911	0.25	16766	0.36
32	24	431	0.01	431	0.00	431	0.01	25	0.00	25	0.00	352	0.74	435	0.00	424	0.00
33	21	1480	0.02	1436	0.01	1391	0.03	446	0.26	22	0.63	947	2.29	2291	0.01	1073	0.01
34	21	223	0.00	223	0.00	223	0.00	22	0.00	22	0.00	158	0.50	227	0.00	216	0.00
35	22	65965	1.43	63186	0.46	68281	1.70	24021	0.83	11113	6.36	7448	8.27	165170	0.63	61548	1.06
36	22	571766	12.62	371834	3.41	458402	11.77	48350	2.98	2783	14.07	188564	111.99	1669788	9.44	717884	18.27
37	23	1307	0.03	1417	0.01	1363	0.03	24	0.02	24	0.01	277	2.10	1532	0.00	1342	0.01
38	13	301	0.01	372	0.00	326	0.01	14	0.01	14	0.01	33	0.74	562	0.00	357	0.00
39	23	2486	0.05	2942	0.02	2682	0.07	24	0.08	24	0.07	146	1.78	4103	0.01	2597	0.02
40	20	259683	8.59	182608	2.70	270195	11.73	38837	1.88	7767	12.86	23371	87.91	1036992	6.74	229210	9.51
41	10	31	0.00	34	0.00	31	0.00	11	0.00	11	0.00	21	0.16	54	0.00	35	0.00
42	30	1855	0.02	1747	0.01	1739	0.02	1117	0.18	31	0.18	1773	1.29	1908	0.01	1636	0.01
43	20	328	0.00	328	0.00	328	0.00	21	0.00	21	0.00	256	0.50	333	0.00	315	0.00
44	19	2990	0.07	3430	0.03	3121	0.08	20	0.05	20	0.05	407	2.18	4142	0.01	3235	0.02
45	20	347	0.00	376	0.00	359	0.01	21	0.01	21	0.00	121	0.74	434	0.00	358	0.00
46	34	60888	0.86	61842	0.31	61563	0.99	36941	0.67	32582	4.05	19865	6.91	80785	0.25	65984	0.63
47	27	4104	0.09	4522	0.03	4284	0.11	28	0.04	28	0.04	515	2.32	5075	0.01	4406	0.02
48	37	12080249	604.43	17435137	247.20	13514084	784.80	129627	2.37	2500	11.08	200559	101.21			19020089	286.02
49	47							2048368	15.84	594399	23.32	2772875	1408.64				
50	23	637	0.01	659	0.01	645	0.02	24	0.02	24	0.02	390	1.40	690	0.00	642	0.00
tpp-ipc5																	
01	5	6	0.00	6	0.00	6	0.00	6	0.00	6	0.00	6	0.01	7	0.00	6	0.00
02	8	9	0.00	11	0.00	9	0.00	9	0.00	9	0.00	9	0.01	26	0.00	16	0.00
03	11	12	0.00	27	0.00	16	0.00	12	0.00	12	0.00	12	0.03	116	0.00	83	0.00
04	14	15	0.00	78	0.00	47	0.00	15	0.01	15	0.00	15	0.07	494	0.00	430	0.00
05	19	623	0.02	5110	0.08	1455	0.05	20	0.36	20	0.77	624	0.48	24698	0.12	17398	0.15
06	25	5843306	179.03	6916518	95.86	6153923	222.35	947059	14.22	74798	23.97					9267024	216.69
trucks-ipc5																	
01	13	1691	0.03	1027	0.01	1039	0.03	14	0.03	14	0.02	285	0.56	5774	0.02	402	0.01
02	17	9624	0.23	2898	0.04	2957	0.11	4192	0.22	18	0.17	1413	1.04	28348	0.14	939	0.03
03	20	80693	2.99	20752	0.44	22236	1.14	199405	2.89	173790	6.88	4049	4.43	379582	2.97	9465	0.40
04	23	1753866	48.55	1205793	23.48	1315672	50.35	2591561	29.17	2568634	56.96	8817	7.75	2990366	26.65	209140	9.43
05	25	12472562	515.50	8007189	242.98	9483222	512.55	23444940	392.99			14744	23.12			1248571	90.78
06	30											308920	343.47				
07	23	2134728	96.15	719751	16.91	755608	50.72	7575415	88.91	8080496	117.13	43270	27.62	12410588	117.92	223011	19.34
08	25			5199440	221.76	6630689	687.95					49663	47.61			3106944	403.36
09	28											233577	248.21				
zenotravel-ipc3																	
01	1	2	0.00	2	0.00	2	0.00	2	0.00	2	0.00	2	0.45	2	0.00	2	0.00
02	6	17	0.00	18	0.00	17	0.00	7	0.00	7	0.00	9	0.46	58	0.00	22	0.00
03	6	28	0.01	18	0.01	12	0.01	7	0.21	7	0.90	40	3.42	5160	0.04	492	0.02
04	8	99	0.01	88	0.01	81	0.01	9	0.20	9	0.89	215	3.44	5256	0.03	665	0.01
05	11	177	0.01	220	0.01	136	0.02	12	0.25	12	1.90	422	7.70	82289	0.63	12466	0.33
06	11	2287	0.10	1144	0.05	504	0.05	12	0.38	12	3.54	1957	11.81	596531	5.90	85931	2.47
07	15	5088	0.16	4234	0.09	4199	0.19	16	0.38	16	3.48	34890	30.36	405626	3.56	115348	2.60
08	11	3268	0.35	1026	0.12	1655	0.32	14354	2.00	12	14.48	83533	292.05			687846	50.76
09	21	2844771	177.70	2842546	176.05	2433822	262.84	2517035	51.18	611457	30.47						
10	22	2283679	295.65	1921903	196.38	1832871	383.99	1322871	34.84	137872	25.44						
11	14	139687	18.63	76904	8.20	93782	19.51	310030	11.28	110726	26.65						

Table B.11: Similar to Table B.7 for the PSR, TPP, TRUCKS, and ZENOTRAVEL domains.

		Forks		Inverted Forks		Both		FA-10^4		FA-10^5		hsp_f		blind		h_{max}	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
schedule-strips																	
02-0	3	5	0.07	5	0.04	5	0.08	4	511.10	4	1743.32	5	577.39	76	0.02	5	0.09
02-1	2	3	0.08	4	0.05	3	0.10	3	104.98			3	754.26	6	0.02	3	0.07
02-2	2	3	0.17	3	0.06	3	0.19	3	231.99			3	495.56	5	0.02	3	0.07
02-3	3	26	0.17	37	0.06	26	0.18	4	56.51			4	658.90	529	0.03	95	0.45
02-4	3	68	0.17	188	0.07	220	0.26					4	484.62	543	0.03	108	0.44
02-5	2	3	0.17	3	0.05	3	0.19	3	363.11			3	667.32	3	0.03	3	0.07
02-6	2	3	0.07	5	0.04	3	0.09	3	121.84			3	697.42	6	0.02	3	0.06
02-7	2	3	0.15	3	0.05	3	0.17	3	323.77			3	604.06	13	0.02	3	0.07
02-8	2	3	0.17	3	0.05	3	0.19	3	316.53			3	668.79	8	0.02	3	0.07
02-9	3	5	0.07	5	0.04	5	0.08	4	251.46			5	577.16	76	0.03	5	0.09
03-0	4	40	0.31	407	0.16	140	0.45							11915	0.60	1127	8.98
03-1	2	3	0.22	3	0.08	3	0.25							31	0.04	25	0.37
03-2	4	27	0.21	50	0.09	33	0.25	5	191.03					3617	0.23	1228	9.56
03-3	4	15	0.13	91	0.09	15	0.15	5	259.13					3379	0.23	170	1.85
03-4	3	4	0.39	16	0.10	4	0.44					4	1223.90	301	0.06	22	0.27
03-5	4	73	0.38	471	0.14	74	0.43	5	682.30					12217	0.64	1175	12.43
03-6	4	72	0.12	75	0.08	69	0.13	5	121.58					2663	0.19	1542	11.73
03-7	4	28	0.23	50	0.09	28	0.25	5	195.72					12859	0.68	1323	13.47
03-8	4	273	0.43	266	0.14	273	0.48							12616	0.65	1590	11.13
03-9	4	8	0.23	31	0.09	14	0.27	5	235.48					4339	0.27	913	7.69
04-0	5	373	0.45	1498	0.50	167	0.54							312193	26.88	22993	273.38
04-1	6	17559	15.45	10707	3.48	17686	17.58	7	1115.76					552069	49.79		
04-2	5	209	0.40	406	0.19	66	0.34							47696	4.97	9703	131.69
04-3	5	142	0.40	674	0.25	251	0.58	6	267.29					89272	8.74	12941	163.84
04-4	5	921	1.14	450	0.31	574	1.39							62013	6.03	13614	168.07
04-5	6	483	0.95	4544	1.11	850	2.11	7	837.68							107978	1399.99
04-6	6	779	0.56	11610	2.44	1834	1.43	7	459.19							107115	1001.40
04-7	5	99	0.58	424	0.31	163	0.78	6	936.68					61327	5.97	8683	103.50
04-8	5	102	0.52	573	0.24	111	0.60	6	711.65					340467	29.56	15122	181.98
04-9	4	1043	1.27	996	0.67	1050	1.66	5	316.22					41673	4.27	5480	83.69
05-0	5	163	0.86	483	0.51	167	1.05							143350	22.71	43336	751.35
05-1	6	2701	2.95	18878	11.36	1257	3.10										
05-2	7	118855	86.65			158640	178.66										
05-3	7	27159	24.88	41447	13.08	13622	16.72										
05-4	6	989	1.63	3433	1.29	582	1.36										
05-5	6	198	0.61	9550	4.61	347	1.05									120602	989.42
05-6	7	6033	11.16	49873	16.17	10325	16.63										
05-7	6	944	1.92	17562	9.03	2107	4.10										
05-8	7	1190	2.43	61539	20.22	2709	7.24										
05-9	6	1537	2.24	15829	6.85	2717	5.45										
06-2	6	888	3.29	26986	22.47	1709	6.91										
06-4	8	11535	20.81			56273	131.69										
06-6	8	15589	46.68			41764	133.76										
07-0	7	2489	9.10			6995	25.49										
07-7	8	10726	41.01			38251	154.49										
07-9	8	6829	19.20			30148	109.49										

Table B.12: Similar to Table B.7 for the (non-IPC) SCHEDULE-STRIPS domain.

		$h^{\mathcal{F}}$		$h^{\mathcal{J}}$		$h^{\mathcal{FJ}}$		HSP^*_{F}		blind	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
elevators-strips-ipc6											
01	42	7483	0.39	10507	0.84	8333	1.03	12935	30.55	26670	0.40
02	26	2898	0.45	5184	1.12	4044	1.46	4810	42.63	16162	0.49
03	55	61649	4.00	219439	13.43	139760	15.62	276441	469.96	650316	11.32
04	40	60039	10.59	294029	74.29	146396	62.26	278087	885.94	1025329	29.51
05	55	909822	68.19	3269854	290.07	2113017	317.53			9567169	174.38
06	53	716238	125.83	3869775	1167.78	1965371	965.39				
11	56	18313	1.34	50734	4.64	31545	5.00	72109	190.25	145170	2.53
12	54	21812	3.39	78362	23.84	46386	21.36	74663	325.43	152021	4.47
13	59	186526	18.43	432280	66.00	297147	68.80			1426461	32.06
14	63	248709	43.80	1325517	337.57	687420	290.86			6238743	199.63
15	66	201777	31.37	2823019	570.43	1255479	425.27				
18	61	1057327	327.82								
21	48	71003	5.99	79574	9.93	66582	13.62	123510	443.99	194669	4.28
22	54	890048	112.11	859710	349.90	757718	395.63			1633295	57.19
23	69	4089071	335.39	10935187	1208.05	7542146	1319.93				
24	56	1430559	291.88								
25	63	1384406	203.57			4430537	1578.04				
26	48	699757	249.28								
openstacks-strips-ipc6											
01	2	209	0.00	209	0.00	209	0.00	49	0.37	193	0.00
02	2	769	0.02	769	0.00	769	0.02	144	0.73	769	0.00
03	2	1729	0.04	1729	0.01	1729	0.04	317	1.32	1665	0.01
04	3	8209	0.17	8209	0.07	8209	0.20	2208	2.63	8113	0.06
05	4	16705	0.41	16705	0.18	16705	0.48	4220	4.87	17151	0.16
06	2	3658	0.11	3658	0.04	3658	0.13	998	5.66	3288	0.02
07	5	195109	5.85	195109	2.45	195109	6.85	61253	40.74	201137	2.31
08	5	228847	7.77	228847	3.23	228847	9.06	70808	57.12	234328	2.92
09	3	116425	5.03	116425	1.61	116425	5.77	4920	18.26	114281	1.32
10	3	77681	3.57	77681	1.10	77681	4.14	5261	23.40	72673	0.76
11	4	575677	28.75	575677	9.11	575677	32.97	98783	105.44	563261	7.17
12	3	354913	19.85	354913	5.63	354913	22.85	10580	43.05	341169	4.11
13	4	2596593	150.86	2596593	46.30	2596593	172.13	398023	443.57	2547985	35.07
14	4	1260363	81.43	1260363	23.36	1260363	93.01	157304	222.14	1233115	17.19
15	4	11995225	867.27	11995225	245.32	11995225	987.24	711526	1034.92	11926297	184.57
16	4	5064737	379.45	5064737	104.37	5064737	432.44	411732	671.53	4928793	75.73
17	4	8193065	673.91	8193065	179.00	8193065	765.15	421646	745.34	8065113	128.80
18	3	1020905	88.15	1020905	22.24	1020905	99.67	34754	186.40	953049	14.32
19	4							812451	1731.49		
21	3							473553	1018.62		
22	4	1805050	204.83	1805050	48.73	1805050	233.98	173929	651.93	1536764	27.62
parcprinter-strips-ipc6											
01	169009	19	0.01	15	0.00	15	0.00	12	0.20	20	0.00
02	438047	240	0.02	183	0.01	179	0.02	19	1.44	1375	0.01
03	807114	880	0.04	821	0.01	668	0.03	334	0.72	4903	0.03
04	876094	142314	13.85	77520	2.28	68116	7.56	993	11.21	12302518	126.46
05	1145132	1780073	219.49	892002	31.78	822442	115.96	6922	35.00		
06	1514200	4113487	613.02	3529327	148.95	3443221	557.75	19613	115.36		
11	182808	25	0.01	24	0.00	24	0.01	10	0.55	23	0.00
12	510256	1183	0.07	1243	0.04	1135	0.08	153	3.39	5138	0.05
13	693064	74201	5.83	144084	4.27	97683	9.25	8348	18.14	1130810	12.72
14	1020512	4491265	463.93					422571	792.78		
21	143411	13	0.00	13	0.00	13	0.00	9	0.09	16	0.00
22	375821	225	0.01	303	0.01	282	0.02	22	0.51	2485	0.02
23	519232	4376	0.28	15825	0.47	8778	0.63	260	2.03	285823	3.32
24	751642	96748	8.49	694503	24.62	316839	31.37	2281	6.38		
25	1215840							68293	145.47		
26	1216460							121897	404.98		
scanalyzer-strips-ipc6											
01	18	19788	1.68	22012	5.20	19809	6.39	21259	13.69	44047	0.68
02	22	37182	1.88	37569	4.36	37524	6.07	29253	13.92	45529	0.54
03	26	43115	1.90	43298	4.02	43298	5.71	37754	14.05	45882	0.49
04	24	3947796	687.38							10175657	314.87
05	30	9193480	870.50							10310817	242.27
06	36	10140909	869.52							10321465	222.66
22	13	46	0.14	51	0.08	46	0.20	6	0.05	54	0.01
23	13	46	0.15	51	0.08	46	0.19	6	0.05	54	0.01
24	13	46	0.14	51	0.08	46	0.19	6	0.05	54	0.01
25	26	8974317	834.36							10170980	113.29
26	30	9936832	720.23							10254740	95.91
27	34	10202674	643.41							10294023	88.03

Table B.13: Runtimes of cost-optimal heuristic-search planners on the ELEVATORS, OPENSTACKS-STRIPS-08, PARCPRINTER, and SCANALYZER domains. The description of the planners is given in Section 4.4.2; here the fork-decomposition heuristics are via databased implicit abstractions. Column *task* denotes problem instance, column h^* denotes optimal solution length. Other columns capture the run *time* and number of expanded *nodes*.

		$h^{\mathcal{F}}$		$h^{\mathcal{J}}$		$h^{\mathcal{FJ}}$		HSP^*_{F}		blind	
task	h^*	nodes	time	nodes	time	nodes	time	nodes	time	nodes	time
pegsol-strips-ipc6											
01	2	12	0.02	10	0.02	10	0.03	6	0.16	11	0.00
02	5	84	0.07	83	0.12	83	0.18	20	5.17	66	0.01
03	4	208	0.07	209	0.12	209	0.19	50	6.91	174	0.00
04	4	193	0.07	181	0.12	181	0.19	15	1.82	192	0.01
05	4	266	0.03	251	0.02	251	0.04	43	5.62	242	0.01
06	4	1343	0.16	901	0.14	901	0.27	247	25.68	1265	0.01
07	3	217	0.08	110	0.12	110	0.18	26	11.67	215	0.01
08	6	31681	2.56	25253	0.71	25253	2.89	7898	28.50	30776	0.15
09	5	3743	0.36	3951	0.20	3951	0.53	757	23.02	3538	0.03
10	6	29756	2.45	28241	0.77	28241	3.21	7522	28.25	29658	0.14
11	7	13832	1.08	12881	0.38	12881	1.36	5979	20.60	13430	0.06
12	8	39340	2.98	37358	0.86	37358	3.63	21133	32.73	38561	0.18
13	9	33379	2.51	33374	0.76	33374	3.09	25897	33.29	32370	0.15
14	7	63096	4.82	55127	1.29	55127	5.63	17144	32.20	62047	0.29
15	8	77932	5.84	73733	1.67	73733	7.09	37810	38.72	76150	0.35
16	8	10491	0.83	10598	0.33	10598	1.10	7939	27.70	10090	0.05
17	10	299676	22.38	300972	6.38	300972	27.00	282810	124.39	294396	1.44
18	7	63247	4.93	50222	1.37	50222	5.55	10358	29.81	62726	0.29
19	8	279822	20.71	257988	5.62	257988	24.49	90950	61.77	275969	1.29
20	7	329570	27.36	293860	8.56	293860	36.43	83693	63.99	328583	1.63
21	8	548254	41.78	494477	11.49	494477	50.51	141906	87.89	545896	2.64
22	6	69922	5.66	48190	1.43	48190	5.95	13123	30.94	69465	0.33
23	8	1262645	97.46	954593	25.16	954593	108.53	181830	114.89	1258767	6.17
24	8	1326517	106.00	1219589	31.83	1219589	136.27	271157	157.50	1324907	6.69
25	8	830637	68.95	899323	25.33	899323	107.61	201932	122.27	830182	4.33
26	9	7196836	553.11	6943124	177.06	6943124	719.81	2031156	1024.04	7178802	37.78
27	7	6092258	523.12	2121936	82.61	2121936	339.83	132701	118.20	6091864	34.53
sokoban-strips-ipc6											
01	11	372	0.03	287	0.01	269	0.02	1079	6.17	1762	0.01
02	9	551	0.02	497	0.01	509	0.02	700	4.57	1348	0.00
03	10	394	0.01	177	0.00	173	0.01	621	2.63	1165	0.00
04	29	130524	5.57	45048	0.28	44198	2.07	282895	177.07	320446	1.43
05	8	50	0.32	203202	7.28	3073	0.96			9607487	81.84
06	9	526	0.04	534	0.01	526	0.04	6815	6.83	10526	0.04
07	15	47522	2.81	42195	0.38	28163	1.84	75669	174.92	315405	1.49
08	31	2114443	135.12	1204212	11.27	1080337	74.55			13329538	77.70
09	19	23083	1.47	26189	0.26	16013	1.12	459188	400.45	818693	4.09
10	30	69797	3.17	21291	0.18	20741	1.07	620685	315.43	852150	4.07
11	35	271598	15.63	282061	2.09	271598	16.87	440869	586.91	531305	2.71
12	32	155166	10.98	60655	0.70	46865	3.69			4705742	25.21
13	20	169436	8.93	294710	3.63	169436	10.26	1631677	994.61	2363177	12.60
14	29	20737	1.05	6984	0.07	6952	0.41	178574	121.96	255203	1.17
15	76	7943562	602.99	7742698	84.75	7456505	622.89			21598353	120.25
16	50	335238	20.20	242778	1.66	240912	15.14	852948	859.44	935561	4.74
17	37	80459	4.17	40425	0.29	36889	2.05	239522	220.86	317984	1.43
18	49	2109516	156.88	119938	0.97	119784	9.18			7219504	39.20
19	47	5238957	354.84	3558809	33.60	3459314	251.38			23255133	130.46
20	2	648	0.14	648	0.69	648	0.79			649	0.01
21	10	337852	74.21	450027	14.64	76647	16.85				
22	44	5866700	473.77	4053413	45.18	3868663	335.31				
23	31	3565151	222.48	3613835	50.31	2563159	181.66				
24	50	14504610	1151.55	2244156	30.10	1759660	154.33				
25	39			23044275	275.91	17832156	1612.04				
26	33			12138101	152.95	10473204	996.25				
27	23					8738457	1131.26				
30	14	2074534	679.61								
transport-strips-ipc6											
01	54	60	0.01	12	0.00	16	0.01	60	0.48	64	0.00
02	131	1558	0.06	874	0.10	998	0.15	1567	6.36	2093	0.01
03	250	380375	10.47	225310	48.69	257608	59.28	380982	274.86	408643	3.69
04	318	3526204	164.35	1462063	714.49	1660874	856.87			4204372	50.69
11	456	135	0.02	111	0.01	103	0.02	135	0.94	164	0.00
12	594	14873	0.37	9976	1.41	11130	1.70	14874	19.85	14796	0.12
13	550	372845	15.07	224986	74.09	246069	89.04	373133	454.55	408449	4.36
21	478	62	0.01	67	0.00	62	0.01	62	0.50	112	0.00
22	632	7544	0.18	4455	0.37	5408	0.54	7544	7.71	7610	0.06
23	630	100269	3.65	56897	13.82	70579	19.32	100347	92.93	106548	1.07
24	614	1587821	77.96	292004	120.98	382588	196.82			1663856	19.29
woodworking-strips-ipc6											
01	170	4313	0.23	3716	0.10	4157	0.28	119	0.85	9086	0.09
02	185	5550	0.34	5054	0.14	5408	0.41	409	3.03	21076	0.30
03	275							80794	136.95		
11	130	860	0.10	987	0.05	897	0.13	50	0.93	3487	0.05
12	225	328229	41.44	328728	16.57	328930	52.03	11665	18.49	1862476	37.91
13	215	4413726	954.34	4125788	455.35	4404104	1297.06	113386	273.76		
21	95	54	0.02	54	0.02	53	0.03	16	0.91	227	0.00
22	185	31189	4.66	67528	3.26	38912	6.83	1931	5.96	177942	3.97
23	195	44641	8.39	155426	9.71	64840	14.42	4673	9.05	962698	23.76

Table B.14: Similar to Table B.13 for the Pegsol, Sokoban, Transport, and Woodworking domains.

B.3 Optimal Cost Partition

		Forks						Inverted Forks						Both					
		Uniform			Optimal			Uniform			Optimal			Uniform			Optimal		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
airport-ipc4																			
01	8	2	10	0.01	3	9	0.08	5	9	0.00	8	9	0.07	3	9	0.00	8	9	0.14
02	9	9	12	0.03	9	11	5.52	4	15	0.01	9	11	0.10	7	15	0.03	9	11	5.82
03	17	10	86	0.25	13	72	43.04	5	133	0.07	17	28	0.53	9	93	0.31	17	28	29.55
04	20	2	22	0.02	3	21	0.50	19	21	0.02	20	21	0.44	7	21	0.02	20	21	1.03
05	21	21	23	1.29	21	22	274.77	8	30	0.06	21	22	0.68	19	27	1.43	21	22	257.34
06	41	22	513	36.72				11	639	1.54	41	43	6.28	22	567	45.25			
07	41	22	514	37.00				11	632	1.53	41	88	9.86	22	550	44.15			
08	62							13	21544	166.51	62	818	176.39						
10	18	2	19	0.02	3	19	1.06	17	19	0.02	18	19	0.54	7	19	0.03	18	19	1.72
11	21	21	23	1.90	21	22	725.11	8	30	0.08	21	22	1.05	19	27	2.13	21	22	765.71
12	39	22	475	54.18				11	728	2.76	39	59	5.91	22	568	71.23			
13	37	20	434	47.48				10	663	2.60	37	64	10.59	20	479	59.82			
14	60							12	25110	334.72	60	81	40.48						
15	58							12	23317	307.60	58	443	97.77						
16	79										79	1038	616.62						
blocks-ipc2																			
04-0	6	6	15	0.01	6	15	0.88	3	46	0.01	6	15	0.10	4	17	0.01	6	15	0.97
04-1	10	4	14	0.01	4	11	0.51	2	31	0.00	4	11	0.06	3	15	0.00	4	11	0.56
04-2	6	6	7	0.01	6	7	0.38	3	26	0.00	6	7	0.05	4	10	0.00	6	7	0.42
05-0	12	6	32	0.03	6	20	2.22	2	302	0.06	6	20	0.21	4	113	0.08	6	20	2.46
05-1	10	6	37	0.03	6	34	4.26	2	280	0.06	6	34	0.39	4	98	0.07	6	34	4.70
05-2	16	7	152	0.09	8	101	12.69	2	596	0.10	8	101	1.24	4	348	0.18	8	101	13.94
06-0	12	9	33	0.04	10	29	20.27	3	766	0.27	10	29	0.56	5	207	0.25	10	29	8.16
06-1	10	9	41	0.07	10	39	45.12	3	2395	0.74	10	39	1.08	5	578	0.78	10	39	17.09
06-2	20	9	855	0.80	10	667	481.26	3	5444	1.23	10	667	12.99	5	3352	2.88	10	667	187.29
07-0	20	10	278	0.56	12	208	916.87	3	20183	8.26	12	208	7.73	5	4022	8.18	12	208	1736.20
07-1	22	9	6910	11.22				2	59207	17.37	10	2269	84.64	5	38539	49.71			
07-2	20	9	1458	2.85				2	46009	15.05	10	376	15.59	5	18854	29.61			
08-0	18	11	1533	4.79				2	344157	179.42	12	389	24.36	5	69830	208.07			
08-1	20	9	10040	27.97				2	517514	236.64	10	3625	189.30	5	191352	475.33			
08-2	16	13	479	1.79				3	237140	136.18	14	221	12.38	6	32567	110.76			
09-1	28	12	3435	18.17	14	635	1101.84				14	635	63.65				14	635	1184.42
09-2	26	14	6379	35.22							16	3022	283.57						
depots-ipc3																			
01	10	4	114	0.24	4	118	109.83	2	279	0.11	6	81	1.44	3	161	0.32	6	81	88.92
02	15	6	1134	10.82				1	9344	12.40	9	527	42.24	4	2638	22.68			
driverlog-ipc3																			
01	7	3	49	0.05	4	32	6.95	3	37	0.01	4	26	0.46	3	37	0.04	4	26	6.58
02	19	12	15713	18.27				11	18452	10.29	13	3930	161.75	12	15794	23.80	15	2464	1644.95
03	12	8	164	0.25	11	17	9.73	8	190	0.13	11	21	0.91	8	163	0.31	11	17	10.59
04	16	11	6161	19.15	14	533	812.63	10	10778	17.14	13	542	65.05	10	7665	29.88	14	382	679.14
05	18	12	13640	45.02	15	273	361.33	12	11400	18.91	14	503	74.02	12	10984	46.16	16	165	294.71
06	11	8	608	5.21	9	156	625.88	7	795	3.60	9	87	28.51	8	492	6.05	10	50	270.77
07	13	11	864	9.56	13	26	143.26	10	1730	7.71	12	36	10.37	11	1006	13.80	13	26	153.80
09	22							12	198651	849.04	15	4123	1690.45						
10	17	12	4304	199.81				12	16099	85.74	16	143	66.78	13	4037	200.52			
11	19	11	43395	1421.90				12	41445	186.53	16	363	285.09	11	39069	1395.51			
freecell-ipc3																			
01	8	5	234	1.54	7	17	82.26	3	974	4.88	8	9	9.32	5	274	3.25	8	9	45.77
02	14	5	30960	107.07				3	75150	230.54				5	37131	224.62			
03	18	6	197647	877.16															
grid-ipc1																			
01	14	4	571	60.28				6	1117	9.49	6	520	979.48	5	472	55.87			
gripper-ipc1																			
01	11	5	214	0.04	5	214	6.33	3	240	0.02	5	214	0.89	4	214	0.05	5	214	7.15
02	17	7	1768	0.54	7	1768	98.40	3	1832	0.36	7	1768	18.19	6	1803	0.75	7	1768	120.37
03	23	9	11626	5.38	9	11626	1077.57	3	11736	4.05	9	11626	250.42	7	11689	8.11	9	11626	1412.66
04	29	11	68380	43.58				3	68558	35.24				8	68479	70.72			
05	35	13	376510	328.10				3	376784	296.59				10	376653	560.93			

Table B.15: Runtimes of cost-optimal heuristic-search planners on the AIRPORT, BLOCKSWORLD, DEPOTS, DRIVERLOG, FREECELL, GRID, and GRIPPER domains. The description of the planners is given in Section 5.5; here the fork-decomposition heuristics are computed fully online. The *task* column denotes problem instance, the h^* column denotes the optimal solution length. Other columns capture the initial evaluation I, number of expanded (*nodes*), and run (*time*).

		Forks						Inverted Forks						Both					
		Uniform			Optimal			Uniform			Optimal			Uniform			Optimal		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
logistics-ipc1																			
01	26										25	89	73.87						
05	22	19	3293	945.35							22	24	12.20						
17	42										42	43	1444.74						
31	13	11	436	9.67	13	14	237.37	10	1981	2.53	13	14	0.98	10	1284	21.84	13	14	243.33
32	20	17	392	2.57	20	21	78.83	15	2704	2.24	19	29	1.26	16	962	5.53	20	21	81.49
33	27				27	28	1171.86				26	1623	308.75				27	28	1229.77
logistics-ipc2																			
04-0	20	20	21	0.02	20	21	4.16	19	193	0.06	20	21	0.22	19	65	0.06	20	21	4.34
04-1	19	19	20	0.03	19	20	3.88	16	570	0.13	19	20	0.22	17	293	0.16	19	20	3.85
04-2	15	15	16	0.02	15	16	2.22	12	117	0.03	15	16	0.11	12	79	0.05	15	16	2.35
05-0	27	27	28	0.05	27	28	8.24	25	2550	0.98	27	28	0.49	25	1171	1.09	27	28	8.69
05-1	17	17	18	0.03	17	18	4.22	14	675	0.19	17	18	0.18	14	427	0.31	17	18	4.39
05-2	8	8	9	0.02	8	9	1.64	7	24	0.01	8	9	0.08	7	13	0.02	8	9	1.78
06-0	25	25	26	0.06	25	26	9.78	22	4249	1.85	25	26	0.51	23	2461	2.54	25	26	10.28
06-1	14	14	15	0.03	14	15	4.39	11	181	0.09	14	15	0.15	12	99	0.13	14	15	4.54
06-2	25	25	26	0.05	25	26	9.52	22	2752	1.22	25	26	0.50	23	1394	1.51	25	26	9.95
06-9	24	24	25	0.04	24	25	7.72	20	2395	0.94	24	25	0.33	21	1428	1.34	24	25	8.04
07-0	36	36	37	0.42	36	37	116.92	31	251287	203.64	36	37	2.09	32	98053	386.80	36	37	126.58
07-1	44	43	1689	10.08							44	45	2.94				44	45	167.13
08-0	31	31	32	0.42	31	32	111.87	26	82476	78.73	31	32	1.57	27	35805	161.33	31	32	121.26
08-1	44	44	45	0.66	44	45	188.02	39	1183608	1306.92	44	45	3.07				44	45	199.99
09-0	36	36	37	0.54	36	37	158.16	30	351538	407.06	36	37	1.90	31	167038	883.68	36	37	157.30
09-1	30	30	31	0.50	30	31	134.35	26	59336	80.88	30	31	1.71	27	25359	168.73	30	31	137.33
10-0	45	45	46	2.26	45	46	746.43				45	46	5.60				45	46	812.66
10-1	42	42	43	2.10	42	43	719.74				42	43	6.41				42	43	756.90
11-0	48	48	697	26.78							48	49	8.06				48	49	1051.09
11-1	60	59	21959	696.23							60	61	13.55				60	61	1557.49
12-0	42	42	43	2.78	42	43	920.83				42	43	7.05				42	43	989.07
12-1	68										68	69	17.24						
miconic-strips-ipc2																			
01-0	4	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01
01-1	3	2	5	0.00	2	5	0.01	2	5	0.00	2	5	0.01	2	5	0.00	2	5	0.01
01-2	4	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01
01-3	4	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01
01-4	4	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01
02-0	7	4	19	0.00	4	19	0.06	3	22	0.00	4	19	0.03	3	19	0.00	4	19	0.08
02-1	7	3	21	0.00	3	21	0.07	2	23	0.00	3	21	0.03	3	21	0.00	3	21	0.09
02-2	7	3	21	0.00	3	21	0.07	2	23	0.00	3	21	0.03	3	21	0.00	3	21	0.09
02-3	7	3	24	0.01	3	24	0.07	3	24	0.00	3	24	0.03	3	24	0.00	3	24	0.09
02-4	7	3	19	0.00	3	19	0.06	2	22	0.00	3	19	0.03	3	19	0.00	3	19	0.08
03-0	10	5	86	0.01	5	86	0.59	3	129	0.01	5	86	0.48	4	98	0.01	5	86	1.05
03-1	11	5	120	0.01	6	113	0.71	3	168	0.01	6	113	0.56	4	147	0.01	6	113	1.26
03-2	10	3	137	0.01	3	137	0.66	2	143	0.01	3	137	0.55	3	137	0.01	3	137	1.18
03-3	10	5	96	0.01	5	90	0.57	3	153	0.01	5	90	0.47	4	117	0.01	5	90	1.04
03-4	10	4	103	0.01	4	103	0.61	3	149	0.01	4	103	0.49	4	115	0.01	4	103	1.09
04-0	14	7	524	0.06	8	449	4.30	4	843	0.08	8	449	6.41	5	686	0.12	8	449	11.42
04-1	13	7	505	0.06	8	419	4.11	4	817	0.08	8	419	6.46	5	663	0.12	8	419	10.48
04-2	15	6	685	0.08	7	608	5.12	4	942	0.09	7	608	7.39	5	802	0.13	7	608	12.79
04-3	15	6	681	0.07	7	604	5.05	4	942	0.09	7	604	7.39	5	798	0.13	7	604	12.74
04-4	15	7	685	0.07	8	608	5.13	4	942	0.09	8	608	7.53	5	802	0.13	8	608	12.76
05-0	17	8	2468	0.37	9	2003	29.33	5	4009	0.66	9	2003	72.60	6	3307	0.93	9	2003	105.21
05-1	17	7	2807	0.42	8	2329	34.37	4	4345	0.71	8	2329	83.43	6	3677	1.01	8	2329	121.00
05-2	15	7	1596	0.29	7	1327	24.89	4	2981	0.55	7	1327	61.96	6	2275	0.73	7	1327	88.97
05-3	17	8	2256	0.36	9	1843	28.93	5	3799	0.62	9	1843	72.71	6	3104	0.87	9	1843	103.45
05-4	18	8	3210	0.46	9	2541	33.72	5	4732	0.78	9	2541	83.01	6	4267	1.11	9	2541	121.00
06-0	19	8	9379	1.98	9	7155	167.35	5	17665	4.74	9	7155	620.45	7	13531	5.90	9	7155	806.61
06-1	19	9	9106	1.93	11	6092	148.43	5	18134	4.75	11	6092	557.51	7	14052	5.94	11	6092	731.35
06-2	20	8	10900	2.19	9	8528	200.04	5	19084	4.90	9	8528	696.50	7	15111	6.28	9	8528	905.40
06-3	20	9	12127	2.43	10	8626	214.68	5	21708	5.69	10	8626	738.41	7	17807	7.19	10	8626	956.47
06-4	21	9	13784	2.62	11	9517	197.24	5	23255	5.93	11	9517	722.80	7	19536	7.66	11	9517	948.46
07-0	23	10	53662	13.29	12	37883	1132.60	6	96092	37.56				8	79449	46.76			
07-1	24	11	56328	13.86	13	38056	1140.83	6	99109	38.56				8	83677	47.49			
07-2	22	11	48141	12.52	13	28170	915.04	6	96139	38.02				8	78471	46.17			
07-3	22	11	46867	12.11	13	28553	925.43	6	93117	36.63				8	75424	44.43			
07-4	25	11	84250	18.24	13	63230	1541.14	6	126595	46.11				8	111984	61.34			
08-0	27	12	272580	81.51				7	485051	267.27				9	408114	317.78			
08-1	27	13	284415	86.93				7	527216	288.07				9	446837	347.43			
08-2	26	11	207931	66.37				7	414294	235.89				9	330993	271.03			
08-3	28	12	369479	104.29				7	598031	320.33				9	527216	392.87			
08-4	27	11	297516	87.65				7	507910	278.64				9	431432	333.91			
09-0	31	13	1461729	497.72															
09-1	30	13	1207894	438.69				7	2335166	1787.13									
09-2	30	13	1294691	460.11				7	2340411	1791.16									
09-3	32	13	1840936	589.09															
09-4	28	13	1252484	467.94															

Table B.16: Similar to Table B.15 for the LOGISTICS-IPC1, LOGISTICS-IPC2, and MICONIC domains.

		Forks						Inverted Forks						Both					
		Uniform			Optimal			Uniform			Optimal			Uniform			Optimal		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
openstacks-ipc5																			
01	23	14	2264	0.49	15	1834	54.33	9	3895	1.19	15	1834	38.50	12	3070	1.36	15	1834	246.77
02	23	14	2617	0.56	15	2140	62.86	9	4485	1.32	15	2140	43.51	12	3561	1.57	15	2140	263.72
03	23	14	2264	0.49	15	1834	54.38	9	3895	1.15	15	1834	38.49	12	3070	1.36	15	1834	259.27
04	23	14	2264	0.49	15	1834	54.37	9	3895	1.15	15	1834	38.58	12	3070	1.36	15	1834	258.62
05	23	14	2264	0.48	15	1834	54.32	9	3895	1.15	15	1834	38.50	12	3070	1.35	15	1834	264.92
06	45	28	366768	255.00				15	779710	1599.86				22	587482	1498.20			
07	46	28	410728	277.99				15	760668	1546.44				22	606782	1515.46			
pathways-ipc5																			
01	6	1	1624	0.03	2	1299	1.34	2	1299	0.02	2	1299	0.66	2	1299	0.03	2	1299	1.59
02	12	2	2755	0.08	4	2307	3.62	4	2307	0.06	4	2307	1.46	3	2437	0.09	4	2307	4.35
03	18	3	44928	2.59	6	20416	53.50	6	20416	1.06	6	20416	20.52	5	29106	2.14	6	20416	67.43
04	17	4	126950	11.45	8	33788	141.43	8	33788	2.97	8	33788	54.82	6	58738	7.07	8	33788	181.58
mprime-ipc1																			
01	5	3	196	0.19	4	24	11.19	5	10	0.03	5	6	0.64	4	24	0.07	5	6	7.08
02	7	4	11604	422.83				3	44045	1620.68	7	13	160.93	5	2565	242.83			
03	4	2	427	35.09				4	7	0.50	4	5	15.39	3	11	3.15	4	5	1267.83
04	8	4	3836	6.62	5	549	230.97	5	1775	1.17	6	144	16.83	5	1093	3.44	7	16	35.21
07	5	2	3314	14.91	4	17	230.86	4	47	0.15	4	17	9.69	3	346	3.07	4	17	249.35
09	8	4	19838	454.91				3	100188	1798.69	7	79	184.28	5	5227	284.13			
11	7	6	9	0.16	7	8	161.42	6	219	0.54	7	8	5.58	6	8	0.16	7	8	159.69
12	6	2	16320	192.10				3	8118	46.69	4	844	275.00	3	5243	95.01	5	22	1322.44
16	6	4	252	171.97							5	35	982.79	4	448	447.49			
17	4							3	453	671.03	4	5	655.98						
25	4	2	75	0.10	4	5	4.77	3	30	0.04	4	5	0.33	3	29	0.08	4	5	5.17
27	5	4	54	2.28	5	6	477.91	3	1772	33.82	5	6	26.54	5	9	1.31	5	6	512.64
28	7	7	8	0.03	7	8	5.88	4	403	0.23	7	8	1.17	6	37	0.08	7	8	7.97
29	4	2	182	4.53	4	5	228.36	3	56	1.11	4	5	7.07	3	32	1.79	4	5	243.11
31	4	2	248	52.86				3	46	7.83	4	6	33.24	3	19	11.79			
32	7	2	31759	133.33				3	12436	34.94	4	3116	870.55	3	11839	95.52	6	67	1200.73
34	4	2	234	11.65				3	46	2.13	4	6	22.85	3	23	3.08	4	5	757.85
35	5	2	392	3.09	3	44	706.09	3	290	2.54	4	25	5.93	3	84	1.89	4	7	159.69
psr-small-ipc4																			
01	8	1	10	0.00	2	9	0.04	1	10	0.00	2	9	0.01	1	10	0.00	2	9	0.04
02	11	1	52	0.01	2	36	0.67	1	55	0.00	2	36	0.13	1	52	0.01	2	36	0.79
03	11	1	31	0.01	2	27	0.25	1	31	0.00	2	27	0.06	1	31	0.00	2	27	0.30
04	10	1	66	0.04	2	28	2.28	1	91	0.03	2	28	0.43	1	73	0.06	2	28	2.74
05	11	1	75	0.01	2	47	1.17	1	79	0.01	2	47	0.26	1	75	0.02	2	47	1.42
06	8	1	10	0.00	2	9	0.04	1	10	0.00	2	9	0.02	1	10	0.00	2	9	0.04
07	11	1	61	0.01	2	48	0.72	1	61	0.00	2	48	0.17	1	61	0.01	2	48	0.86
08	8	1	24	0.01	2	11	0.36	1	29	0.00	2	11	0.08	1	25	0.01	2	11	0.43
09	8	1	18	0.01	2	9	0.17	1	19	0.00	2	9	0.04	1	18	0.00	2	9	0.19
10	7	2	131	0.20	3	44	8.13	2	183	0.18	3	44	1.64	2	155	0.32	3	44	10.28
11	19	2	149	0.03	4	141	3.72	2	149	0.02	4	141	0.71	2	149	0.04	4	141	4.42
12	16	2	120	0.03	4	83	2.59	2	123	0.02	4	83	0.56	2	120	0.04	4	83	3.15
13	15	3	90	0.02	5	82	1.64	3	90	0.01	5	82	0.34	3	90	0.02	5	82	1.96
14	9	2	19	0.00	3	13	0.17	2	19	0.00	3	13	0.04	2	19	0.00	3	13	0.19
15	10	2	1200	6.55	2	62	71.41	2	708	6.25	2	62	5.02	2	769	9.91	2	62	79.57
16	25	2	2328	0.65	3	1961	97.64	2	2158	0.34	3	1961	9.57	2	2176	0.85	3	1961	111.58
17	9	2	15	0.00	3	13	0.09	2	15	0.00	3	13	0.02	2	15	0.00	3	13	0.10
18	12	2	85	0.03	3	56	1.73	2	90	0.01	3	56	0.36	2	85	0.03	3	56	2.05
19	25	3	8025	4.31	5	6934	735.96	3	7856	2.19	5	6934	52.31	2	7876	5.80	5	6934	823.43
20	17	3	80	0.02	5	74	1.62	3	80	0.01	5	74	0.35	3	80	0.02	5	74	1.95
21	10	3	28	0.01	4	21	0.34	3	28	0.00	4	21	0.08	3	28	0.01	4	21	0.41
22	33	3	163299	405.65				3	176058	245.42				3	168685	617.45			
23	12	3	77	0.04	4	46	3.01	3	93	0.03	4	46	0.57	3	77	0.06	4	46	3.61
24	10	3	28	0.01	4	21	0.34	3	28	0.00	4	21	0.08	3	28	0.01	4	21	0.40
25	9	2	485	84.24				2	463	145.38	2	74	107.79	2	482	213.42			
26	17	3	144	0.05	5	101	4.09	3	150	0.03	5	101	0.85	3	146	0.06	5	101	4.98
27	21	3	616	0.33	5	380	37.99	3	675	0.21	5	380	6.35	3	650	0.49	5	380	45.39
28	14	3	79	0.02	4	71	2.01	3	79	0.01	4	71	0.37	3	79	0.02	4	71	2.37
29	21	4	142772	436.34				3	187319	307.77	9	11733	1778.17	4	159325	709.89			
30	22	3	1791	1.25	5	976	135.40	3	1982	0.80	5	976	22.84	3	1883	1.90	5	976	164.19
31	19	3	11278	25.93	4	3027	1303.42	3	6810	38.66	4	3027	134.35	3	8297	53.43	4	3027	1515.10
32	24	4	431	0.17	6	421	24.01	4	431	0.10	6	421	4.25	4	431	0.25	6	421	31.65
33	21	3	1480	0.84	4	894	99.28	2	1436	0.30	4	894	6.95	2	1391	1.00	4	894	112.08
34	21	4	223	0.07	6	213	8.77	4	223	0.04	6	213	1.65	4	223	0.09	6	213	10.50
35	22	3	65965	160.36				2	63186	39.55	4	15841	407.66	2	68281	199.30			
36	22	4	571766	392.49				5	371834	786.06				5	458402	1094.61			
37	23	5	1307	1.29	9	749	155.51	5	1417	0.95	9	749	28.46	5	1363	2.10	9	749	191.64
38	13	3	301	0.20	5	84	11.05	3	372	0.15	5	84	2.07	3	326	0.32	5	84	13.42
39	23	3	2486	2.49	5	1113	241.32	3	2942	1.64	5	1113	35.91	3	2682	3.91	5	1113	281.20
40	20							2	182608	1384.90	3	17410	1634.22						
41	10	3	31	0.01	4	22	0.47	3	34	0.00	4	22	0.11	3	31	0.01	4	22	0.56
42	30	3	1855	0.50	4	1701	80.79	2	1747	0.17	4	1701	7.07	2	1739	0.59	4	1701	92.19
43	20	4	328	0.09	6	312	11.63	4	328	0.05	6	312	2.06	4	328	0.12	6	312	13.81
44	19	4	2990	3.25	8	1103	271.15	4	3430	2.30	8	1103	46.50	4	3121	5.24	8	1103	327.14
45	20	4	347	0.16	6	232	18.72	4	376	0.11	6	232	3.29	4	359	0.25	6	232	21.50
46	34	2	60888	51.77				2	61842	21.14	5	46436	498.90	2	61563	68.33			
47	27	3	4104	5.27	7	1803	530.37	3	4522	3.93	7	1803	96.45	3	4284	8.70	7	1803	647.01
50	23	6	637	0.39	8	552	53.58	6	659	0.26	8	552	9.89	6	645	0.60	8	552	65.78

Table B.17: Similar to Table B.15 for the Openstacks, Pathways, Mprime, and PSR domains.

		Forks						Inverted Forks						Both					
		Uniform			Optimal			Uniform			Optimal			Uniform			Optimal		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
mystery-ipc1																			
01	5	3	7	0.01	4	6	0.87	5	6	0.00	5	6	0.07	4	6	0.01	5	6	1.62
02	7	4	2404	64.94				3	8012	234.10	7	13	67.70	5	722	47.50	7	8	1209.10
03	4	2	73	1.92	3	10	498.22	4	7	0.12	4	5	2.59	3	11	0.59	4	5	246.47
09	8	4	3049	47.68				3	10764	137.61	7	47	57.63	5	1215	40.75	8	9	1042.00
11	7	6	9	0.02	7	8	14.24	6	33	0.03	7	8	0.49	6	8	0.02	7	8	20.46
17	4	2	354	200.98				3	85	26.31	4	5	80.60	3	83	90.17			
19	6							2	4968	183.24	3	518	1163.55						
25	4	2	9	0.02	4	5	2.00	3	10	0.01	4	5	0.12	3	9	0.02	4	5	2.14
26	6	2	1807	50.40				2	1835	25.34	3	413	186.77	2	1344	60.20			
27	5	4	14	0.27	5	6	109.09	3	159	1.61	5	6	4.94	5	6	0.22	5	6	116.82
28	7	7	8	0.01	7	8	1.77	4	47	0.02	7	8	0.28	6	15	0.02	7	8	2.96
29	4	2	31	0.26	4	5	39.31	3	14	0.10	4	5	1.17	3	10	0.17	4	5	42.15
pipesworld-notankage-ipc4																			
01	5	1	121	0.15	2	42	59.70	1	109	0.05	2	42	2.90	1	121	0.18	2	42	105.74
02	12	2	1413	2.05				2	1542	0.86	4	945	72.54	2	1413	2.42			
03	8	2	1742	5.26				1	3001	3.31	3	567	149.52	2	1742	6.43			
04	11	3	7007	24.71				2	8911	12.43	5	2487	885.06	3	7007	30.79			
05	8	2	4093	27.45				2	6805	19.74	4	647	696.11	2	4093	35.40			
06	10	3	12401	105.37				2	27377	103.75				3	12401	140.53			
07	8	3	4370	71.75				2	9168	68.10	5	338	1311.81	3	4370	105.53			
08	10	4	18851	406.67				3	56189	483.28				4	20584	600.94			
11	20							3	472950	1577.22									
13	16							4	117475	899.72									
21	14	4	23833	1663.46				3	49035	495.53									
pipesworld-tankage-ipc4																			
01	5	1	77	0.13	2	43	102.62	1	126	0.07	2	43	2.55	1	105	0.20	2	43	132.92
02	12	2	960	1.20				2	1005	0.60	4	770	45.09	2	960	1.55			
03	8	2	20803	155.53				1	52139	158.91				2	20803	207.57			
04	11	3	110284	1004.10				2	157722	668.67				3	110284	1408.50			
05	8	2	6531	73.63				1	13148	79.04	4	857	1787.43	2	6531	112.61			
06	10	3	20171	329.40				2	43583	310.24				3	20171	460.45			
rovers-ipc5																			
01	10	6	147	0.01	6	147	1.07	6	147	0.01	6	147	0.37	6	147	0.02	6	147	1.35
02	8	6	44	0.01	6	44	0.53	6	44	0.01	6	44	0.17	6	44	0.01	6	44	0.70
03	11	5	672	0.11	6	419	5.02	6	419	0.05	6	419	1.37	6	448	0.10	6	419	6.15
04	8	6	47	0.02	7	20	0.41	7	20	0.00	7	20	0.13	6	24	0.01	7	20	0.50
05	22	11	808084	237.13				14	410712	123.64				13	522937	231.28			
07	18							10	741649	517.18				8	1682245	1780.27			
satellite-ipc4																			
01	9	6	24	0.00	7	16	0.21	6	32	0.00	7	16	0.23	6	29	0.00	7	16	0.44
02	13	10	86	0.02	11	24	0.82	8	337	0.10	11	24	1.03	8	241	0.13	11	24	1.87
03	11	5	2249	1.24	9	77	9.45	7	656	0.53	9	66	15.18	7	728	0.82	9	66	23.56
04	17	10	9817	10.65	16	204	26.39	11	14860	24.90	16	157	81.88	11	11250	26.18	16	157	96.47
05	15	7	279569	1251.83				10	46453	515.80	13	345	1232.42	9	61692	877.26	13	345	1775.28
06	20	10	1496577	968.24				10	1572327	1721.87									
tpp-ipc5																			
01	5	5	6	0.00	5	6	0.03	4	6	0.00	5	6	0.01	5	6	0.00	5	6	0.03
02	8	8	9	0.00	8	9	0.08	7	11	0.00	8	9	0.02	8	9	0.00	8	9	0.08
03	11	11	12	0.00	11	12	0.20	9	27	0.00	11	12	0.04	10	16	0.00	11	12	0.20
04	14	14	15	0.01	14	15	0.40	11	78	0.01	14	15	0.06	13	47	0.01	14	15	0.42
05	19	15	623	0.52	19	20	4.80	13	5110	1.36	17	21	0.38	15	1455	1.21	19	20	5.31
trucks-ipc5																			
01	13	5	1691	0.41	7	1043	31.71	6	1027	0.22	7	1013	18.87	6	1039	0.40	7	1013	55.89
02	17	7	9624	2.68	9	4309	185.92	9	2898	0.57	10	2898	91.12	8	2957	1.35	10	2898	260.71
03	20	8	80693	71.37				11	20752	19.93	12	19568	1771.56	10	22236	31.25			
04	23	8	1753866	1237.60				11	1205793	850.34				9	1315672	1394.88			
07	23	10	2134728	1313.60				13	719751	408.75				11	755608	820.55			
zenotravel-ipc3																			
01	1	1	2	0.01	1	2	0.11	1	2	0.00	1	2	0.07	1	2	0.01	1	2	0.18
02	6	4	17	0.02	5	9	0.34	3	18	0.02	5	9	0.21	4	17	0.02	5	9	0.55
03	6	4	28	0.08	6	7	5.05	5	18	0.12	6	7	1.01	5	12	0.11	6	7	6.23
04	8	5	99	0.15	8	9	4.54	5	88	0.26	7	13	1.64	5	81	0.30	8	9	6.57
05	11	8	177	0.32	11	12	13.20	9	220	0.22	11	12	3.06	9	136	0.36	11	12	17.48
06	11	8	2287	5.51	11	12	30.95	9	1144	2.00	11	12	6.02	9	504	2.40	11	12	39.07
07	15	8	5088	9.63	12	608	452.82	9	4234	5.56	12	782	164.96	9	4199	10.58	12	608	648.39
08	11	7	3268	43.96				8	1026	8.92	10	82	203.39	8	1655	30.06	11	61	974.44
10	22				21	24	912.61				21	66	765.08				22	23	1455.83
11	14							10	76904	1090.67	13	82	762.36						

Table B.18: Similar to Table B.15 for the Mystery, Pipesworld-NoTankage, Pipesworld-Tankage, Rovers, Satellite, TPP, Trucks, and Zenotravel domains.

		Forks						Inverted Forks						Both					
		Uniform			Optimal			Uniform			Optimal			Uniform			Optimal		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
schedule-strips																			
02-0	3	2	5	0.15	2	5	10.42	2	5	0.14	2	5	20.51	2	5	0.22	2	5	33.48
02-1	2	2	3	0.16	2	3	17.62	2	4	0.11	2	3	5.25	2	3	0.18	2	3	23.88
02-2	2	2	3	0.32	2	3	54.18	2	3	0.17	2	3	7.53	2	3	0.40			
02-3	3	2	26	0.50				2	37	0.76	2	27	69.04	2	26	0.61			
02-4	3	2	68	1.34				2	188	2.24	2	27	73.48	1	220	7.20			
02-5	2	2	3	0.33	2	3	45.69	2	3	0.14	2	3	6.30	2	3	0.38			
02-6	2	2	3	0.14	2	3	16.20	2	5	0.12	2	3	4.67	2	3	0.17	2	3	22.04
02-7	2	2	3	0.30	2	3	41.00	2	3	0.13	2	3	5.81	2	3	0.34			
02-8	2	2	3	0.32	2	3	56.07	2	3	0.14	2	3	6.63	2	3	0.38			
02-9	3	2	5	0.15	2	5	10.42	2	5	0.14	2	5	20.49	2	5	0.22	2	5	33.44
03-0	4	3	40	2.72	3	28	566.96	2	407	12.16	3	28	87.94	2	140	14.55			
03-1	2	2	3	0.51	2	3	108.90	2	3	0.35	2	3	20.83	2	3	0.72	2	3	1324.64
03-2	4	3	27	1.16	3	26	511.15	3	50	1.83	3	26	77.77	3	33	2.33			
03-3	4	3	15	0.79	3	15	138.40	2	91	2.39	3	15	38.58	3	15	0.96	3	15	185.64
03-4	3	3	4	1.11	3	4	251.00	2	16	2.08	3	4	25.80	3	4	1.52	3	4	314.01
03-5	4	3	73	6.13				2	471	16.71	3	32	130.28	3	74	8.32			
03-6	4	3	72	1.27	4	5	46.61	2	75	1.80	3	26	103.58	3	69	1.33	4	5	82.13
03-7	4	3	28	1.05				3	50	1.83	3	28	98.93	3	28	1.43			
03-8	4	3	273	11.53				2	266	11.46	3	54	257.27	3	273	17.48			
03-9	4	3	8	0.96	3	6	199.91	3	31	1.77	3	6	30.46	3	14	2.13			
04-0	5	4	373	13.91	5	6	287.85	3	1498	74.46	4	65	377.48	3	167	24.60			
04-1	6	3	17559	1373.80				3	10707	626.54	4	89	1563.71						
04-2	5	4	209	9.88	5	6	145.78	3	406	20.85	4	36	259.88	4	66	5.30			
04-3	5	3	142	10.47	5	6	287.67	3	674	33.29	4	11	123.53	3	251	29.28			
04-4	5	4	921	64.48				3	450	46.95	4	211	1529.21	3	574	116.65			
04-5	6	4	483	47.25				3	4544	268.77	4	125	1675.86	3	850	187.46			
04-6	6	4	779	27.09	5	44	502.80	3	11610	361.74	4	237	1759.53	3	1834	102.68	5	44	783.34
04-7	5	3	99	18.48	4	7	494.08	3	424	38.04	4	7	84.60	3	163	40.04			
04-8	5	3	102	16.01	5	6	289.10	3	573	31.87	4	29	125.14	3	111	23.35			
04-9	4	2	1043	80.06	4	6	335.74	2	996	76.64	3	78	1116.60	2	1050	143.48			
05-0	5	3	163	41.61	5	6	664.67	3	483	63.23	4	9	198.49	3	167	62.53			
05-1	6	5	2701	213.92	6	7	726.57							4	1257	286.28			
05-3	7				6	16	1148.62							4	13622	1693.68			
05-4	6	4	989	100.02				4	3433	229.05	5	114	715.41	4	582	100.05			
05-5	6	5	198	21.67				3	9550	767.94	5	198	1179.47	4	347	68.64			
05-6	7	4	6033	743.61	6	35	1314.44							4	10325	1508.56			
05-7	6	4	944	131.19				3	17562	1446.20	5	99	517.72	4	2107	379.70			
05-8	7	5	1190	172.59										4	2709	730.54			
05-9	6	4	1537	140.49	5	34	1725.99	3	15829	1248.19				3	2717	547.56			
06-2	6	4	888	243.14										4	1709	730.36			
06-4	8	6	11535	1776.87															
07-0	7	5	2489	786.76							6	149	1703.36						
07-9	8	6	6829	1559.86															

Table B.19: Similar to Table B.15 for the non-IPC SCHEDULE-STRIPS domain.

B.4 Beyond Optimal Cost Partition

		Forks						Inverted Forks						Both					
		Optimal for I			Uniform			Optimal for I			Uniform			Optimal for I			Uniform		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
airport-ipc4																			
01	8	3	10	0.01	2	10	0.01	8	9	0.01	5	9	0.00	8	9	0.02	3	9	0.00
02	9	9	12	0.41	9	12	0.01	9	12	0.02	4	15	0.00	9	12	0.53	7	15	0.01
03	17	13	84	0.67	10	86	0.02	17	34	0.04	5	133	0.01	17	34	0.69	9	93	0.02
04	20	3	22	0.04	2	22	0.01	20	21	0.04	19	21	0.00	20	21	0.06	7	21	0.01
05	21	21	23	0.08	21	23	0.08	21	23	0.08	8	30	0.02	19	27	0.09	19	27	0.09
06	41	22	513	0.16	22	513	0.16	41	46	0.13	11	639	0.06	22	567	0.19	22	567	0.19
07	41	22	514	0.15	22	514	0.15	41	114	0.13	11	632	0.05	22	550	0.19	22	550	0.19
08	62	42	12733	1.89	42	12733	1.89	62	1610	0.35	13	21544	1.36	39	14398	4.02	39	14398	4.02
09	71	44	88670	16.58	44	88670	16.58	71	21933	1.76	14	136717	9.60	41	90412	38.78	41	90412	38.78
10	18	3	19	0.07	2	19	0.01	18	19	0.05	17	19	0.01	18	19	0.10	7	19	0.01
11	21	21	23	0.10	21	23	0.10	21	23	0.11	8	30	0.03	19	27	0.12	19	27	0.12
12	39	22	475	0.20	22	475	0.20	39	116	0.17	11	728	0.07	22	568	0.25	22	568	0.25
13	37	20	434	0.20	20	434	0.20	37	81	0.17	10	663	0.07	20	479	0.24	20	479	0.24
14	60	42	12040	2.90	42	12040	2.90	60	1128	0.43	12	25110	1.86	39	15948	4.64	39	15948	4.64
15	58	40	11477	2.74	40	11477	2.74	58	1790	0.45	12	23317	1.71	37	14557	4.25	37	14557	4.25
16	79	61	267277	77.39	61	267277	77.39	79	27067	3.74	13	824491	97.12	55	353592	114.58	55	353592	114.58
17	88	62	2460667	708.82	62	2460667	708.82	88	63766	9.39				57	2678689	1235.79	57	2678689	1235.79
18	107							107	1077502	215.00									
19	90	62	1354353	592.53	62	1354353	592.53	90	112721	18.32	14	3400142	492.06	57	1462739	660.17	57	1462739	660.17
21	101	54	5156	48.29	54	5156	48.29	17	11259	3.72	17	11259	3.72	55	4773	51.13	55	4773	51.13
22	148	55	606648	1110.09	55	606648	1110.09	24	1063668	318.90	24	1063668	318.90	60	477836	1082.91	60	477836	1082.91
36	109	61	9504	129.73	61	9504	129.73	17	34986	14.41	17	34986	14.41	61	9436	140.75	61	9436	140.75
37	142	140	37873	820.33	140	37873	820.33												
blocks-ipc2																			
04-0	6	6	15	0.04	6	15	0.00	6	15	0.02	3	46	0.00	6	15	0.04	4	17	0.00
04-1	10	4	32	0.04	4	14	0.00	4	19	0.02	2	31	0.00	4	19	0.04	3	15	0.00
04-2	6	6	7	0.04	6	7	0.00	6	7	0.02	3	26	0.00	6	7	0.05	4	10	0.00
05-0	12	6	71	0.08	6	32	0.00	6	49	0.03	2	302	0.01	6	49	0.10	4	113	0.00
05-1	10	6	110	0.09	6	37	0.00	6	78	0.02	2	280	0.00	6	78	0.09	4	98	0.00
05-2	16	8	126	0.09	7	152	0.00	8	101	0.02	2	596	0.00	8	101	0.10	4	348	0.01
06-0	12	10	29	0.84	9	33	0.00	10	29	0.04	3	766	0.01	10	33	0.19	5	207	0.01
06-1	10	10	43	0.81	9	41	0.00	10	39	0.04	3	2395	0.03	10	39	0.20	5	578	0.02
06-2	20	10	667	0.44	9	855	0.01	10	667	0.05	3	5444	0.05	10	723	0.22	5	3352	0.06
07-0	20	12	267	2.42	10	278	0.01	12	283	0.05	3	20183	0.28	12	208	5.51	5	4022	0.12
07-1	22	10	4504	1.21	9	6910	0.10	10	17174	0.18	2	59207	0.60	10	4201	2.20	5	38539	0.67
07-2	20	10	860	1.78	9	1458	0.02	10	6920	0.11	2	46009	0.52	10	376	3.92	5	18854	0.39
08-0	18	12	1009	9.34	11	1533	0.03	12	2588	0.10	2	344157	5.46	12	389	26.55	5	69830	2.09
08-1	20	10	10436	6.74	9	10040	0.17	10	56998	0.59	2	517514	7.22	10	3625	21.84	5	191352	4.91
08-2	16	14	237	15.03	13	479	0.02	14	156	0.07	3	237140	4.08	14	221	26.08	6	32567	1.09
09-0	30	14	1658811	31.68	13	134185	3.10	14	402738	5.56	2	7405904	117.14	14	1118398	28.56	6	4346535	118.23
09-1	28	14	172891	4.88	12	3435	0.09	14	28366	0.48	2	4145371	77.54	14	33507	2.05	6	917197	33.32
09-2	26	16	33366	1.81	14	6379	0.17	16	3499	0.14	3	4145278	78.21	16	13957	1.48	7	923365	33.79
10-0	34	18	5340501	126.98	16	1524599	36.52	18	1649448	27.53				18	3755808	122.10			
10-1	32	18	3385326	88.33	16	610206	15.79	18	411390	7.22				18	3577561	123.10			
10-2	34	18	6112602	146.38	16	1516087	37.71	18	2803577	44.86				18	1975227	69.93			
depots-ipc3																			
01	10	4	167	0.53	4	114	0.01	6	167	0.03	2	279	0.01	6	181	0.62	3	161	0.02
02	15	6	2851	3.08	6	1134	0.08	9	2062	0.15	1	9344	0.31	9	2056	3.58	4	2638	0.22
03	27	12	266167	36.70	12	134428	8.59	16	161249	8.56	2	2520703	159.84	16	231645	46.32	7	581726	66.43
04	30	12	1254545	101.18	12	1254545	101.18	13	1402766	35.15				8	5835295	923.87	8	5835295	923.87
07	21	10	198055	29.80	10	109765	9.17	13	99051	7.68	2	4271196	336.59	13	159363	43.92	7	487961	76.02
10	24	8	2964635	283.55	8	2964635	283.55	12	9191303	561.88				6	6081478	1187.66	6	6081478	1187.66
13	25	12	1003709	152.30	12	1003709	152.30	16	1570265	85.59				8	8161872	1559.21	8	8161872	1559.21
driverlog-ipc3																			
01	7	4	64	0.08	3	49	0.00	4	64	0.02	3	37	0.00	4	64	0.10	3	37	0.00
02	19	15	10546	0.66	12	15713	0.42	13	18682	0.26	11	18452	0.27	15	12178	0.82	12	15794	0.55
03	12	11	25	0.14	8	164	0.00	11	21	0.03	8	190	0.00	11	25	0.14	8	163	0.01
04	16	14	5204	0.75	11	6161	0.42	13	9080	0.29	10	10778	0.30	14	5109	0.91	10	7665	0.62
05	18	15	2724	0.53	12	13640	1.01	14	15883	0.53	12	11400	0.36	16	881	0.48	12	10984	1.07
06	11	9	701	0.56	8	608	0.09	9	973	0.11	7	795	0.06	10	339	0.75	8	492	0.11
07	13	13	78	0.65	11	864	0.14	12	980	0.11	10	1730	0.11	13	96	0.73	11	1006	0.21
08	22	15	4842469	469.23	12	669994	75.74	15	955110	45.31	13	1181268	61.32	17	1362122	194.45	13	694996	104.59
09	22	16	137555	21.70	12	150255	14.72	15	133658	6.83	12	198651	11.44	16	71722	15.74	12	164109	23.06
10	17	16	417	6.65	12	4304	0.44	16	1242	0.24	12	16099	1.21	16	957	6.15	13	4037	0.69
11	19	15	16993	8.48	11	43395	4.99	16	7054	0.77	12	41445	2.22	16	4002	10.55	11	39069	5.90
13	26	15	1303099	325.71	15	1303099	325.71	21	76356	15.72	15	1014865	144.64	15	1098694	422.20	15	1098694	422.20
14	28							23	542561	98.36									

Table B.20: Runtimes of cost-optimal heuristic-search planners on the AIRPORT, BLOCKSWORLD, DEPOTS, and DRIVERLOG domains. The description of the planners is given in Section 5.5; here the fork-decomposition heuristics are via implicit abstraction databases. The *task* column denotes the problem instance, the h^* column denotes the optimal solution length. Other columns capture the initial evaluation (I), number of expanded (*nodes*), and run (*time*).

		Forks						Inverted Forks						Both					
		Optimal for I			Uniform			Optimal for I			Uniform			Optimal for I			Uniform		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
freecell-ipc3																			
01	8	7	705	1.93	5	234	0.10	8	541	0.34	3	974	0.15	8	303	1.35	5	274	0.17
02	14	5	30960	1.95	5	30960	1.95	9	42549	4.01	3	75150	5.53	5	37131	4.79	5	37131	4.79
03	18	6	197647	14.41	6	197647	14.41	11	280303	37.57	3	533995	78.27	6	240161	51.24	6	240161	51.24
04	26	8	1427669	97.22	6	997836	60.67	8	1894652	130.01	3	1921470	232.95	6	1218329	213.02	6	1218329	213.02
05	30	8	7369373	557.79	5	6510089	448.22							8	8712420	987.04			
logistics-ipc2																			
04-0	20	20	21	0.05	20	21	0.00	20	21	0.02	19	193	0.00	20	21	0.05	19	65	0.00
04-1	19	19	20	0.04	19	20	0.00	19	20	0.01	16	570	0.01	19	20	0.04	17	293	0.00
04-2	15	15	16	0.04	15	16	0.00	15	16	0.01	12	117	0.00	15	16	0.04	12	79	0.00
05-0	27	27	28	0.06	27	28	0.00	27	28	0.02	25	2550	0.05	27	28	0.06	25	1171	0.03
05-1	17	17	18	0.05	17	18	0.00	17	18	0.01	14	675	0.01	17	18	0.05	14	427	0.01
05-2	8	8	9	0.04	8	9	0.00	8	9	0.01	7	24	0.00	8	9	0.04	7	13	0.00
06-0	25	25	26	0.06	25	26	0.00	25	26	0.02	22	4249	0.09	25	26	0.07	23	2461	0.07
06-1	14	14	15	0.05	14	15	0.00	14	15	0.02	11	181	0.00	14	15	0.06	12	99	0.00
06-2	25	25	26	0.06	25	26	0.00	25	169	0.02	22	2752	0.06	25	187	0.08	23	1394	0.04
06-9	24	24	25	0.06	24	25	0.00	24	25	0.02	20	2395	0.04	24	25	0.06	21	1428	0.04
07-0	36	36	37	0.38	36	37	0.00	36	8726	0.39	31	251287	7.52	36	1480	0.55	32	98053	4.59
07-1	44	43	1689	0.46	43	1689	0.07	44	291	0.05	37	3532213	99.33	44	291	0.54	38	1705009	72.35
08-0	31	31	32	0.41	31	32	0.00	31	347	0.04	26	82476	2.69	31	32	0.43	27	35805	1.78
08-1	44	44	45	0.46	44	45	0.01	44	17554	0.98	39	1183608	45.72	44	45	0.52	40	462244	25.36
09-0	36	36	37	0.47	36	37	0.00	36	1296	0.08	30	351538	13.75	36	1206	0.59	31	167038	9.76
09-1	30	30	31	0.43	30	31	0.00	30	211	0.03	26	59336	2.48	30	31	0.44	27	25359	1.73
10-0	45	45	3421	1.72	45	46	0.01	45	7547	0.55				45	3621	2.11			
10-1	42	42	253	1.52	42	43	0.01	42	8723	0.63				42	43	1.81			
11-0	48	48	4279	2.17	48	697	0.09	48	11082	0.93				48	20667	5.30			
11-1	60	59	20523	3.77	59	21959	2.22	60	630873	62.97				60	54749	12.98			
12-0	42	42	2041	1.92	42	43	0.02	42	14795	1.12				42	7137	3.01			
12-1	68	67	106534	13.46	67	106534	11.64												
13-1	64	63	214096	105.22															
14-0	58	57	1720905	534.37															
miconic-strips-ipc2																			
01-0	4	1	5	0.00	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01	1	5	0.00
01-1	3	2	5	0.00	2	5	0.00	2	5	0.01	2	5	0.00	2	5	0.01	2	5	0.00
01-2	4	1	5	0.00	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01	1	5	0.00
01-3	4	1	5	0.00	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.01	1	5	0.00
01-4	4	1	5	0.01	1	5	0.00	1	5	0.01	1	5	0.00	1	5	0.00	1	5	0.00
02-0	7	4	19	0.01	4	19	0.00	4	19	0.01	3	22	0.00	4	19	0.01	3	19	0.00
02-1	7	3	28	0.01	3	21	0.00	3	21	0.01	2	23	0.00	3	21	0.01	3	21	0.00
02-2	7	3	28	0.01	3	21	0.00	3	25	0.01	2	23	0.00	3	28	0.01	3	21	0.00
02-3	7	3	29	0.01	3	24	0.00	3	26	0.01	3	24	0.00	3	24	0.01	3	24	0.00
02-4	7	3	26	0.01	3	19	0.00	3	19	0.02	2	22	0.00	3	23	0.01	3	19	0.00
03-0	10	5	86	0.01	5	86	0.00	5	96	0.01	3	129	0.00	5	86	0.02	4	98	0.00
03-1	11	6	113	0.01	5	120	0.00	6	113	0.02	3	168	0.00	6	113	0.02	4	147	0.00
03-2	10	3	167	0.01	3	137	0.00	3	167	0.01	2	143	0.00	3	167	0.02	3	137	0.00
03-3	10	5	116	0.01	5	96	0.00	5	116	0.02	3	153	0.00	5	116	0.02	4	117	0.00
03-4	10	4	152	0.01	4	103	0.00	4	152	0.01	3	149	0.00	4	176	0.02	4	115	0.00
04-0	14	8	449	0.02	7	524	0.00	8	449	0.03	4	843	0.00	8	449	0.03	5	686	0.01
04-1	13	8	419	0.02	7	505	0.00	8	419	0.02	4	817	0.00	8	419	0.03	5	663	0.01
04-2	15	7	746	0.02	6	685	0.00	7	746	0.04	4	942	0.00	7	746	0.04	5	802	0.01
04-3	15	7	756	0.02	6	681	0.00	7	756	0.03	4	942	0.00	7	756	0.03	5	798	0.01
04-4	15	8	608	0.02	7	685	0.00	8	608	0.04	4	942	0.00	8	608	0.04	5	802	0.01
05-0	17	9	2003	0.05	8	2468	0.03	9	2255	0.07	5	4009	0.03	9	2003	0.09	6	3307	0.05
05-1	17	8	3221	0.07	7	2807	0.04	8	3221	0.08	4	4345	0.03	8	3221	0.11	6	3677	0.06
05-2	15	7	2055	0.06	7	1596	0.02	7	2401	0.07	4	2981	0.02	7	2055	0.09	6	2275	0.04
05-3	17	9	1843	0.05	8	2256	0.03	9	2059	0.07	5	3799	0.03	9	1843	0.09	6	3104	0.05
05-4	18	9	3164	0.07	8	3210	0.04	9	3164	0.08	5	4732	0.03	9	3164	0.12	6	4267	0.06
06-0	19	9	9637	0.24	8	9379	0.18	9	11061	0.22	5	17665	0.15	9	9637	0.35	7	13531	0.26
06-1	19	11	6092	0.17	9	9106	0.17	11	6871	0.17	5	18134	0.15	11	6092	0.27	7	14052	0.27
06-2	20	9	10805	0.26	8	10900	0.20	9	12948	0.25	5	19084	0.16	9	10805	0.38	7	15111	0.28
06-3	20	10	12534	0.30	9	12127	0.23	10	12534	0.24	5	21708	0.18	10	12534	0.42	7	17807	0.33
06-4	21	11	12269	0.28	9	13784	0.24	11	12269	0.24	5	23255	0.19	11	12269	0.40	7	19536	0.35
07-0	23	12	37883	1.08	10	53662	1.19	12	46181	0.77	6	96092	0.97	12	37883	1.37	8	79449	1.76
07-1	24	13	38056	1.08	11	56328	1.24	13	41531	0.72	6	99109	0.96	13	38056	1.37	8	83677	1.83
07-2	22	13	28170	0.86	11	48141	1.10	13	32174	0.60	6	96139	0.94	13	28170	1.14	8	78471	1.77
07-3	22	13	28553	0.85	11	46867	1.08	13	31944	0.61	6	93117	0.92	13	28553	1.12	8	75424	1.69
07-4	25	13	63230	1.54	11	84250	1.70	13	68778	1.02	6	126595	1.22	13	63230	1.91	8	111984	2.36
08-0	27	14	182583	5.87	12	272580	7.05	14	217443	3.53	7	485051	5.51	14	182583	6.68	9	408114	10.53
08-1	27	16	160709	5.53	13	284415	7.56	16	160709	2.79	7	527216	6.01	16	160709	6.30	9	446837	11.58
08-2	26	13	174001	5.82	11	207931	5.60	13	198548	3.34	7	414294	4.79	13	174001	6.64	9	330993	8.90
08-3	28	15	291195	8.87	12	369479	9.25	15	291195	4.66	7	598031	6.74	15	291195	9.90	9	527216	13.30
08-4	27	13	256333	7.91	11	297516	7.74	13	286425	4.59	7	507910	5.79	13	256333	8.90	9	431432	11.04
09-0	31	15	1178547	42.62	13	1461729	43.82	15	1318542	23.07	7	2491975	32.67	15	1178547	46.10	10	2138656	63.58
09-1	30	16	852612	33.17	13	1207894	37.47	16	930890	16.93	7	2335166	30.76	16	852612	36.20	10	1952916	59.39
09-2	30	15	1014390	37.85	13	1294691	40.03	15	1137842	20.42	7	2340411	30.97	15	1014390	41.37	10	1972234	59.25
09-3	32	16	1458281	49.88	13	1840936	52.68	16	1560206	26.36	7	2889342	38.12	16	1458281	53.85	10	2571844	74.47
09-4	28	15	808604	33.00	13	1252484	40.34	15	1016784	18.93	7	2352633	31.35	15	808604	36.35	10	1944297	59.37
10-0	33	17	3573644	156.43	14	5716041	202.37	17	4384570	86.73	8	10316603	153.80	17	3573644	168.99	11	8774563	300.08
10-1	32	18	3048758	140.35	15	5601282	201.43	18	3712295	75.10	8	10789013	162.69	18	3048758	151.53	11	9144153	315.23
10-2	32	17	2680052	127.22	14	4153191	155.86	17	3241896	67.08	8	9148616	138.69	17	2680052	138.35	11	7466572	265.86
10-3	34	18	3658921	161.91	15	6108094	214.68	18	4158162	82.59	8	10960203	167.10	18	3658921	173.25	11	9400386	320.13
10-4	33	17	4134141	181.00	14	5920127	211.40	17	4864687	97.61	8	11075136	170.82	17	4134141	195.68	11	9448049	322.74
11-0	37	20	15335602	777.41				20	16574717	402.63				20	15335602	832.55			
11-1	34	19	7132395	406.69	16	15349953	668.77	19	9228364	217.74				19	7132395	439.05			
11-4	35	18	11409090	615.93				18	14888659	345.88				18	11409090	662.75			

Table B.21: Similar to Table B.20 for the Freecell, Logistics-ipc2, and Miconic domains.

		Forks						Inverted Forks						Both					
		Optimal for I			Uniform			Optimal for I			Uniform			Optimal for I			Uniform		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
logistics-ipc1																			
01	26	25	214	9.90	22	77763	7.14	25	27570	2.71	20	1469610	95.49	26	2234	12.94	20	830292	98.59
05	22	22	23	11.16	19	3293	0.46	22	15645	1.27	15	850312	42.43	22	527	12.96	17	173477	18.19
31	13	13	35	1.50	11	436	0.03	13	239	0.02	10	1981	0.07	13	158	1.61	10	1284	0.09
32	20	20	21	0.51	17	392	0.01	19	1939	0.08	15	2704	0.07	20	603	0.55	16	962	0.05
33	27	27	28	3.07	23	312180	27.19	26	1609589	119.68				27	28	3.93	21	3617185	427.52
35	30	26	477883	183.08	26	477883	183.08												
grid-ipc1																			
01	14	4	571	0.60	4	571	0.60	6	1117	1.21	6	1117	0.34	5	472	0.78	5	472	0.78
02	26	6	3330274	1078.55	6	3330274	1078.55	11	976834	158.19									
gripper-ipc1																			
01	11	5	214	0.04	5	214	0.00	5	222	0.02	3	240	0.00	5	222	0.04	4	214	0.00
02	17	7	1768	0.10	7	1768	0.02	7	1792	0.04	3	1832	0.01	7	1792	0.13	6	1803	0.03
03	23	9	11626	0.33	9	11626	0.19	9	11674	0.14	3	11736	0.08	9	11674	0.40	7	11689	0.22
04	29	11	68380	1.75	11	68380	1.46	11	68460	0.64	3	68558	0.51	11	68460	2.00	8	68479	1.63
05	35	13	376510	10.76	13	376510	10.07	13	376630	3.53	3	376784	3.20	13	376630	12.02	10	376653	11.11
06	41	15	1982032	75.90	15	1982032	70.91	15	1982200	20.28	3	1982408	19.08	15	1982200	83.62	11	1982227	77.81
07	47	17	10091986	469.24	17	10091986	438.41	17	10092210	111.05	3	10092464	105.67	17	10092210	510.07	12	10092241	478.67
mprime-ipc1																			
01	5	4	28	0.07	3	196	0.02	5	11	0.04	5	10	0.01	5	6	0.08	4	24	0.01
02	7	6	855	11.89	4	11604	2.72	7	43	0.58	3	44045	80.68	7	33	27.48	5	2565	4.20
03	4	3	18	5.81	2	427	0.27	4	7	0.17	4	7	0.08	4	17	5.65	3	11	0.16
04	8	5	1553	0.20	4	3836	0.22	6	144	0.05	5	1775	0.10	7	18	0.11	5	1093	0.09
05	11	6	1076368	211.43	4	1745027	195.08	7	272745	190.03				5	604756	592.60	5	604756	592.60
07	5	4	133	0.59	2	3314	0.25	4	45	0.07	4	47	0.03	4	785	0.75	3	346	0.08
08	6	2	485381	491.53	2	485381	491.53				2	1376780	1426.21						
09	8	6	16391	22.06	4	19838	2.92	7	393	0.88	3	100188	74.85	8	9	26.60	5	5227	6.31
11	7	7	9	0.18	6	9	0.02	7	8	0.07	6	219	0.03	7	9	0.22	6	8	0.03
12	6	3	2042	2.68	2	16320	1.89	4	3818	1.20	3	8118	0.73	5	1405	4.26	3	5243	1.13
16	6	4	252	0.76	4	252	0.76	5	115	3.58	2	51590	135.00	4	448	2.76	4	448	2.76
17	4	2	2746	10.47	2	2746	10.47	4	5	4.22	3	453	18.78	3	451	21.40	3	451	21.40
19	6	2	727401	521.78	2	727401	521.78	3	69185	704.84	2	95361	485.79						
21	6	2	174221	55.09	2	174221	55.09	3	25650	29.57	2	34022	47.43	2	169400	392.30	2	169400	392.30
25	4	4	5	0.12	2	75	0.01	4	6	0.02	3	30	0.01	4	6	0.13	3	29	0.01
26	6	2	77622	24.69	2	77622	24.69	3	18430	46.22	2	147854	48.25	2	68239	106.35	2	68239	106.35
27	5	5	30	2.64	4	54	0.16	5	9	0.31	3	1772	1.50	5	60	2.85	5	9	0.18
28	7	7	8	0.07	7	8	0.01	7	11	0.04	4	403	0.02	7	8	0.08	6	37	0.02
29	4	4	7	1.33	2	182	0.12	4	13	0.15	3	56	0.08	4	15	1.43	3	32	0.11
30	9							7	69337	373.03									
31	4	2	248	0.51	2	248	0.51	4	9	1.05	3	46	0.68	3	19	1.00	3	19	1.00
32	7	4	2499	1.43	2	31759	1.73	4	5337	0.75	3	12436	1.46	6	151	1.76	3	11839	1.93
34	4	3	29	8.37	2	234	0.26	4	29	0.48	3	46	0.16	4	6	6.53	3	23	0.28
35	5	3	269	1.83	2	392	0.07	4	178	0.14	3	290	0.06	4	74	2.01	3	84	0.08
mystery-ipc1																			
01	5	4	7	0.05	3	7	0.00	5	6	0.02	5	6	0.00	5	6	0.06	4	6	0.00
02	7	6	391	11.60	4	2404	0.50	7	27	0.55	3	8012	11.19	7	17	27.94	5	722	1.01
03	4	3	10	5.24	2	73	0.08	4	7	0.13	4	7	0.04	4	7	5.79	3	11	0.10
09	8	6	819	14.22	4	3049	0.37	7	242	0.53	3	10764	5.66	8	10	25.24	5	1215	1.01
11	7	7	8	0.16	6	9	0.01	7	8	0.05	6	33	0.01	7	8	0.20	6	8	0.01
15	6	2	28271	20.21	2	28271	20.21	4	18128	442.29	3	21572	41.22	3	5079	44.42	3	5079	44.42
17	4	2	354	1.32	2	354	1.32	4	5	2.46	3	85	2.74	3	83	3.59	3	83	3.59
19	6	2	21717	4.87	2	21717	4.87	3	4299	4.40	2	4968	5.26	2	16276	29.28	2	16276	29.28
20	7	2	89887	46.32	2	89887	46.32	4	65015	652.92	3	84572	153.53	3	53114	173.34	3	53114	173.34
25	4	4	5	0.12	2	9	0.00	4	5	0.02	3	10	0.00	4	5	0.12	3	9	0.01
26	6	3	2011	15.86	2	1807	0.27	3	590	0.57	2	1835	0.30	4	526	21.54	2	1344	0.69
27	5	5	6	2.65	4	14	0.05	5	6	0.18	3	159	0.09	5	6	3.73	5	6	0.07
28	7	7	8	0.05	7	8	0.00	7	8	0.03	4	47	0.00	7	8	0.07	6	15	0.00
29	4	4	11	1.25	2	31	0.04	4	13	0.09	3	14	0.03	4	27	1.43	3	10	0.06
30	9	4	23175	5.16	4	23175	5.16	7	3986	9.02	3	76480	169.86	5	7232	13.30	5	7232	13.30
openstacks-ipc5																			
01	23	15	1834	0.05	14	2264	0.02	15	1834	0.06	9	3895	0.03	15	1834	0.09	12	3070	0.05
02	23	15	2140	0.06	14	2617	0.03	15	2140	0.06	9	4485	0.04	15	2140	0.10	12	3561	0.05
03	23	15	1834	0.06	14	2264	0.02	15	1834	0.06	9	3895	0.03	15	1834	0.10	12	3070	0.05
04	23	15	1834	0.06	14	2264	0.02	15	1834	0.05	9	3895	0.03	15	1834	0.10	12	3070	0.05
05	23	15	1834	0.06	14	2264	0.02	15	1834	0.06	9	3895	0.03	15	1834	0.10	12	3070	0.05
06	45	30	303840	8.13	28	366768	7.52	30	303840	7.87	15	779710	18.93	30	303840	13.11	22	587482	22.20
07	46	30	350647	9.35	28	410728	8.23	30	350647	10.62	15	760668	18.33	30	350647	14.23	22	606782	22.53
pathways-ipc5																			
01	6	2	1299	0.01	1	1624	0.00	2	1299	0.02	2	1299	0.00	2	1299	0.02	2	1299	0.00
02	12	4	2307	0.03	2	2755	0.02	4	2307	0.03	4	2307	0.01	4	2307	0.03	3	2437	0.02
03	18	6	20566	0.34	3	44928	0.62	6	20416	0.31	6	20416	0.25	6	20566	0.36	5	29106	0.43
04	17	8	33806	0.80	4	126950	2.66	8	33788	0.70	8	33788	0.59	8	33788	0.85	6	58738	1.31

Table B.22: Similar to Table B.20 for the Logistics-ipc1, Grid, Gripper, Mprime, Mystery, Openstacks, and Pathways domains.

		Forks						Inverted Forks						Both					
		Optimal for I			Uniform			Optimal for I			Uniform			Optimal for I			Uniform		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
pipesworld-notankage-ipc4																			
01	5	2	49	0.23	1	121	0.02	2	49	0.06	1	109	0.01	2	49	0.27	1	121	0.02
02	12	4	971	0.55	2	1413	0.06	4	971	0.09	2	1542	0.02	4	971	0.58	2	1413	0.08
03	8	3	597	44.69	2	1742	0.14	3	597	0.14	1	3001	0.07	2	1742	0.18	2	1742	0.18
04	11	3	7007	0.45	3	7007	0.45	5	2626	0.25	2	8911	0.22	3	7007	0.59	3	7007	0.59
05	8	2	4093	0.49	2	4093	0.49	4	659	0.36	2	6805	0.26	2	4093	0.65	2	4093	0.65
06	10	3	12401	1.44	3	12401	1.44	6	1620	0.56	2	27377	1.34	3	12401	2.03	3	12401	2.03
07	8	5	354	12.36	3	4370	0.97	5	354	0.84	2	9168	0.77	5	354	22.19	3	4370	1.34
08	10	7	655	50.39	4	18851	3.84	7	655	1.54	3	56189	6.21	4	20584	6.42	4	20584	6.42
09	13	6	141888	72.28	3	1092472	160.71	6	141888	5.61	2	2419903	151.99	6	141888	72.69	3	1092472	219.75
10	18							8	3038645	106.19									
11	20	4	313952	27.68	4	313952	27.68	5	395194	7.26	3	472950	29.55	4	313952	43.90	4	313952	43.90
12	24	6	684234	75.72	6	684234	75.72	8	927951	56.67	3	1319980	133.58	6	686186	145.41	6	686186	145.41
13	16	6	39998	6.02	6	39998	6.02	9	24928	6.39	4	117475	18.08	6	40226	12.69	6	40226	12.69
15	26	4	1594863	254.43	4	1594863	254.43	8	806802	20.48	3	2588849	192.90	4	1594863	353.40	4	1594863	353.40
17	22	7	5437393	1588.68	7	5437393	1588.68	10	6777356	1278.16									
21	14	4	23833	4.02	4	23833	4.02	7	9653	2.77	3	49035	7.76	4	23833	7.87	4	23833	7.87
23	18	5	2285790	568.93	5	2285790	568.93	9	543995	44.82	3	7047138	871.03	5	2282678	843.28	5	2282678	843.28
24	24							13	6329286	1706.86									
41	12	3	502308	370.68	3	502308	370.68							3	502308	1092.50	3	502308	1092.50
pipesworld-tankage-ipc4																			
01	5	2	50	0.23	1	77	0.02	2	50	0.05	1	126	0.01	2	50	0.28	1	105	0.02
02	12	4	797	0.50	2	960	0.05	4	797	0.09	2	1005	0.02	4	797	0.61	2	960	0.06
03	8	3	4955	3.45	2	20803	1.89	3	4955	0.51	1	52139	2.46	3	4955	4.55	2	20803	2.82
04	11	5	32363	14.55	3	110284	8.06	5	32363	1.72	2	157722	9.60	5	32363	16.72	3	110284	14.05
05	8	4	918	5.48	2	6531	0.86	4	918	0.68	1	13148	1.03	4	918	10.21	2	6531	1.32
06	10	6	2592	17.43	3	20171	2.41	6	2592	1.09	2	43583	4.32	6	2592	35.27	3	20171	4.41
07	8	3	202706	73.83	3	202706	73.83	6	3913	4.43	2	2643752	1379.11	3	202706	208.81	3	202706	208.81
08	11							8	119303	33.16									
11	22	2	2345399	296.87	2	2345399	296.87	6	1971143	180.59	2	2629204	662.94	2	2365735	838.85	2	2365735	838.85
15	30	4	9652091	1721.67	4	9652091	1721.67												
21	14	3	839847	250.39	3	839847	250.39	7	166057	460.68									
31	39	3	1501847	240.38	3	1501847	240.38	7	1411887	386.35	2	1568963	661.88	3	1504072	850.16	3	1504072	850.16
psr-small-ipc4																			
01	8	2	11	0.01	1	10	0.00	2	10	0.01	1	10	0.00	2	10	0.01	1	10	0.00
02	11	2	68	0.01	1	52	0.00	2	70	0.01	1	55	0.00	2	60	0.02	1	52	0.00
03	11	2	31	0.01	1	31	0.00	2	33	0.02	1	31	0.00	2	29	0.01	1	31	0.00
04	10	2	373	0.03	1	66	0.00	2	373	0.03	1	91	0.00	2	293	0.04	1	73	0.00
05	11	2	149	0.02	1	75	0.00	2	149	0.02	1	79	0.00	2	112	0.02	1	75	0.00
06	8	2	11	0.01	1	10	0.00	2	10	0.02	1	10	0.00	2	10	0.01	1	10	0.00
07	11	2	97	0.01	1	61	0.00	2	112	0.01	1	61	0.00	2	81	0.02	1	61	0.00
08	8	2	126	0.01	1	24	0.00	2	131	0.02	1	29	0.00	2	52	0.02	1	25	0.00
09	8	2	43	0.01	1	18	0.00	2	44	0.01	1	19	0.00	2	25	0.02	1	18	0.00
10	7	3	404	0.05	2	131	0.01	3	404	0.04	2	183	0.00	3	286	0.08	2	155	0.01
11	19	4	150	0.02	2	149	0.00	4	153	0.02	2	149	0.00	4	150	0.02	2	149	0.00
12	16	4	146	0.02	2	120	0.00	4	150	0.02	2	123	0.00	4	146	0.03	2	120	0.00
13	15	5	91	0.02	3	90	0.00	5	91	0.01	3	90	0.00	5	89	0.02	3	90	0.00
14	9	3	28	0.02	2	19	0.00	3	19	0.01	2	19	0.00	3	27	0.02	2	19	0.00
15	10	2	3186	0.44	2	1200	0.08	2	748	0.07	2	708	0.03	2	3186	0.48	2	769	0.09
16	25	3	2538	0.05	2	2328	0.02	3	2113	0.03	2	2158	0.01	3	2625	0.06	2	2176	0.03
17	9	3	16	0.01	2	15	0.00	3	14	0.01	2	15	0.00	3	16	0.01	2	15	0.00
18	12	3	149	0.02	2	85	0.00	3	149	0.02	2	90	0.00	3	137	0.03	2	85	0.00
19	25	5	8423	0.16	3	8025	0.11	5	7756	0.08	3	7856	0.05	5	8817	0.18	2	7876	0.12
20	17	5	82	0.02	3	80	0.00	5	84	0.02	3	80	0.00	5	82	0.03	3	80	0.00
21	10	4	41	0.02	3	28	0.00	4	39	0.01	3	28	0.00	4	39	0.02	3	28	0.00
22	33	8	184124	4.11	3	163299	4.17	8	189114	1.70	3	176058	1.56	8	184124	5.08	3	168685	5.01
23	12	4	220	0.04	3	77	0.00	4	220	0.03	3	93	0.00	4	178	0.05	3	77	0.00
24	10	4	41	0.02	3	28	0.00	4	39	0.02	3	28	0.00	4	39	0.02	3	28	0.00
25	9	2	6026	9.21	2	485	3.06	2	740	1.03	2	463	0.58	2	946	9.91	2	482	3.28
26	17	5	173	0.03	3	144	0.00	5	179	0.02	3	150	0.00	5	173	0.04	3	146	0.00
27	21	5	705	0.05	3	616	0.01	5	821	0.02	3	675	0.00	5	705	0.07	3	650	0.01
28	14	4	93	0.02	3	79	0.00	4	93	0.01	3	79	0.00	4	88	0.03	3	79	0.00
29	21	9	139469	4.30	4	142772	4.55	9	188291	2.25	3	187319	2.12	9	139469	5.30	4	159325	5.80
30	22	5	2292	0.09	3	1791	0.03	5	2373	0.06	3	1982	0.01	5	2292	0.12	3	1883	0.04
31	19	4	20865	0.69	3	11278	0.25	4	16392	0.23	3	6810	0.08	4	22357	0.84	3	8297	0.24
32	24	6	431	0.05	4	431	0.01	6	435	0.03	4	431	0.00	6	431	0.06	4	431	0.01
33	21	4	2291	0.07	3	1480	0.02	4	1585	0.03	2	1436	0.01	4	2319	0.09	2	1391	0.03
34	21	6	224	0.03	4	223	0.00	6	227	0.02	4	223	0.00	6	224	0.05	4	223	0.00
35	22	4	146628	2.67	3	65965	1.43	4	87454	0.64	2	63186	0.46	4	128963	2.66	2	68281	1.70
36	22	6	807912	20.06	4	571766	12.62	6	563624	6.03	5	371834	3.41	6	563624	16.94	5	458402	11.77
37	23	9	1263	0.15	5	1307	0.03	9	1498	0.06	5	1417	0.01	9	1263	0.20	5	1363	0.03
38	13	5	307	0.05	3	301	0.01	5	493	0.02	3	372	0.00	5	307	0.07	3	326	0.01
39	23	5	3088	0.12	3	2486	0.05	5	3955	0.06	3	2942	0.02	5	3088	0.16	3	2682	0.07
40	20	3	821479	22.51	2	259683	8.59	3	209183	3.66	2	182608	2.70	3	419694	15.18	2	270195	11.73
41	10	4	61	0.02	3	31	0.00	4	61	0.01	3	34	0.00	4	51	0.03	3	31	0.00
42	30	4	1878	0.04	3	1855	0.02	4	1791	0.02	2	1747	0.01	4	1956	0.04	2	1739	0.02
43	20	6	329	0.03	4	328	0.00	6	329	0.02	4	328	0.00	6	326	0.04	4	328	0.00
44	19	8	2757	0.16	4	2990	0.07	8	3555	0.07	4	3430	0.03	8	2757	0.22	4	3121	0.08
45	20	6	398	0.05	4	347	0.00	6	436	0.03	4	376	0.00	6	398	0.07	4	359	0.01
46	34	5	73394	1.02	2	60888	0.86	5	61630	0.33	2	61842	0.31	5	76463	1.17	2	61563	0.99
47	27	7	4238	0.22	3	4104	0.09	7	5100	0.10	3	4522	0.03	7	4238	0.30	3	4284	0.11
48	37	9	16134714	673.78	4	12080249	604.43				3	17435137	247.20	9	16134714	822.99	4	13514084	784.80
50	23	8	681	0.08	6	637	0.01	8	687	0.04	6	659	0.01	8	681	0.11	6	645	0.02

Table B.23: Similar to Table B.20 for the Pipesworld-NoTankage, Pipesworld-Tankage, and PSR domains.

		Forks						Inverted Forks						Both					
		Optimal for I			Uniform			Optimal for I			Uniform			Optimal for I			Uniform		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
rovers-ipc5																			
01	10	6	173	0.01	6	147	0.00	6	173	0.01	6	147	0.00	6	173	0.01	6	147	0.00
02	8	6	50	0.01	6	44	0.00	6	50	0.02	6	44	0.00	6	50	0.02	6	44	0.00
03	11	6	523	0.02	5	672	0.01	6	494	0.01	6	419	0.00	6	523	0.02	6	448	0.01
04	8	7	20	0.01	6	47	0.00	7	21	0.01	7	20	0.00	7	20	0.01	6	24	0.00
05	22	14	587122	18.75	11	808084	22.61	14	518811	12.12	14	410712	9.23	14	587122	20.90	13	522937	18.29
07	18	11	602274	37.95	6	4546797	191.34	12	396969	11.57	10	741649	21.01	12	401806	30.56	8	1682245	102.77
12	19	12	698692	72.13				12	679360	37.10	11	1529551	76.46	12	698692	79.13			
satellite-ipc4																			
01	9	7	29	0.01	6	24	0.00	7	29	0.02	6	32	0.00	7	29	0.02	6	29	0.00
02	13	11	86	0.02	10	86	0.00	11	91	0.03	8	337	0.00	11	94	0.04	8	241	0.01
03	11	9	767	0.07	5	2249	0.08	9	981	0.07	7	656	0.01	9	1038	0.13	7	728	0.04
04	17	16	1457	0.23	10	9817	0.57	16	1380	0.28	11	14860	0.38	16	1085	0.41	11	11250	0.76
05	15	10	63761	9.98	7	279569	49.47	13	6906	0.98	10	46453	4.92	13	4169	1.67	9	61692	18.85
06	20	15	378078	38.13	10	1496577	92.22	16	228672	10.38	10	1572327	51.68	16	171620	21.08	10	1518261	105.65
07	21							18	2511377	265.04				18	2078741	686.02			
tpp-ipc5																			
01	5	5	6	0.01	5	6	0.00	5	6	0.01	4	6	0.00	5	6	0.01	5	6	0.00
02	8	8	9	0.01	8	9	0.00	8	9	0.01	7	11	0.00	8	9	0.01	8	9	0.00
03	11	11	12	0.02	11	12	0.00	11	12	0.01	9	27	0.00	11	12	0.02	10	16	0.00
04	14	14	15	0.02	14	15	0.00	14	15	0.02	11	78	0.00	14	15	0.02	13	47	0.00
05	19	19	20	0.12	15	623	0.02	17	1306	0.05	13	5110	0.08	19	20	0.13	15	1455	0.05
06	25	7	5843306	201.49	7	5843306	179.03	7	5843306	81.32	5	6916518	95.86	7	5843306	215.08	6	6153923	222.35
trucks-ipc5																			
01	13	7	2043	0.07	5	1691	0.03	7	1013	0.05	6	1027	0.01	7	1117	0.08	6	1039	0.03
02	17	9	11681	0.33	7	9624	0.23	10	2898	0.07	9	2898	0.04	10	2942	0.18	8	2957	0.11
03	20	11	105292	4.45	8	80693	2.99	12	19568	0.59	11	20752	0.44	12	21443	1.42	10	22236	1.14
04	23	13	1699483	56.87	8	1753866	48.55	13	1036115	24.56	11	1205793	23.48	13	1363663	59.56	9	1315672	50.35
05	25	15	11461967	607.64	9	12472562	515.50	15	6172038	225.61	13	8007189	242.98	15	10107973	599.86	10	9483222	512.55
07	23	13	3237871	166.29	10	2134728	96.15	14	626947	17.47	13	719751	16.91	14	763790	61.29	11	755608	50.72
08	25							16	4012227	204.98	14	5199440	221.76	16	6341279	786.44	13	6630689	687.95
zenotravel-ipc3																			
01	1	1	2	0.03	1	2	0.00	1	2	0.03	1	2	0.00	1	2	0.05	1	2	0.00
02	6	5	9	0.03	4	17	0.00	5	9	0.03	3	18	0.00	5	9	0.05	4	17	0.00
03	6	6	83	0.07	4	28	0.01	6	53	0.03	5	18	0.01	6	25	0.09	5	12	0.01
04	8	8	12	0.10	5	99	0.01	7	44	0.06	5	88	0.01	8	10	0.15	5	81	0.01
05	11	11	20	0.14	8	177	0.01	11	105	0.07	9	220	0.01	11	88	0.21	9	136	0.02
06	11	11	634	0.21	8	2287	0.10	11	1830	0.13	9	1144	0.05	11	444	0.30	9	504	0.05
07	15	12	2979	0.27	8	5088	0.16	12	2806	0.13	9	4234	0.09	12	3240	0.45	9	4199	0.19
08	11	10	57	0.99	7	3268	0.35	10	632	0.42	8	1026	0.12	11	119	1.71	8	1655	0.32
09	21	19	84749	12.58	14	2844771	177.70	18	966627	68.56	15	2842546	176.05	20	73137	22.02	15	2433822	262.84
10	22	21	110557	34.92	17	2283679	295.65	21	527430	56.23	18	1921903	196.38	22	13316	13.23	18	1832871	383.99
11	14	13	621	2.35	9	139687	18.63	13	45924	5.47	10	76904	8.20	13	1046	4.45	9	93782	19.51
12	21	19	374833	127.85										20	111037	69.66			
13	26	24	425408	229.06										24	984912	714.39			

Table B.24: Similar to Table B.20 for the Rovers, Satellite, TPP, Trucks, and Zenotravel domains.

		Forks						Inverted Forks						Both					
		Optimal for I			Uniform			Optimal for I			Uniform			Optimal for I			Uniform		
task	h^*	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time	I	nodes	time
schedule-strips																			
02-0	3	2	259	0.23	2	5	0.07	2	5	0.30	2	5	0.04	2	5	0.51	2	5	0.08
02-1	2	2	71	0.40	2	3	0.08	2	3	0.16	2	4	0.05	2	32	0.55	2	3	0.10
02-2	2	2	50	1.16	2	3	0.17	2	3	0.24	2	3	0.06	2	3	0.19	2	3	0.19
02-3	3	2	55	1.06	2	26	0.17	2	141	0.21	2	37	0.06	2	73	1.29	2	26	0.18
02-4	3	2	31	1.22	2	68	0.17	2	370	0.25	2	188	0.07	2	31	1.36	1	220	0.26
02-5	2	2	3	1.10	2	3	0.17	2	6	0.19	2	3	0.05	2	3	0.19	2	3	0.19
02-6	2	2	68	0.37	2	3	0.07	2	7	0.16	2	5	0.04	2	29	0.49	2	3	0.09
02-7	2	2	3	0.97	2	3	0.15	2	3	0.18	2	3	0.05	2	3	39.92	2	3	0.17
02-8	2	2	79	1.08	2	3	0.17	2	9	0.20	2	3	0.05	2	3	0.19	2	3	0.19
02-9	3	2	259	0.23	2	5	0.07	2	5	0.30	2	5	0.04	2	5	0.49	2	5	0.08
03-0	4	3	192	2.01	3	40	0.31	3	956	0.43	2	407	0.16	2	140	0.45	2	140	0.45
03-1	2	2	3	1.80	2	3	0.22	2	19	0.41	2	3	0.08	2	3	1.82	2	3	0.25
03-2	4	3	70	1.29	3	27	0.21	3	138	0.29	3	50	0.09	3	33	0.25	3	33	0.25
03-3	4	3	24169	3.30	3	15	0.13	3	91	0.26	2	91	0.09	3	2254	1.19	3	15	0.15
03-4	3	3	1408	3.59	3	4	0.39	3	4	0.38	2	16	0.10	3	110	4.22	3	4	0.44
03-5	4	4	15	2.19	3	73	0.38	3	233	0.37	2	471	0.14	4	15	2.52	3	74	0.43
03-6	4	4	31	0.48	3	72	0.12	3	30	0.34	2	75	0.08	4	31	0.79	3	69	0.13
03-7	4	3	323	1.31	3	28	0.23	3	204	0.40	3	50	0.09	3	28	0.25	3	28	0.25
03-8	4	3	273	0.43	3	273	0.43	3	318	0.44	2	266	0.14	3	273	0.48	3	273	0.48
03-9	4	3	12	1.33	3	8	0.23	3	77	0.30	3	31	0.09	3	14	0.27	3	14	0.27
04-0	5	5	1320	2.70	4	373	0.45	4	302	0.53	3	1498	0.50	5	970	2.93	3	167	0.54
04-1	6	5	99	2.94	3	17559	15.45	4	38863	6.93	3	10707	3.48	3	17686	17.58	3	17686	17.58
04-2	5	5	10	1.23	4	209	0.40	4	979	0.72	3	406	0.19	4	66	0.34	4	66	0.34
04-3	5	5	1017	2.57	3	142	0.40	4	161	0.48	3	674	0.25	5	681	2.68	3	251	0.58
04-4	5	4	595	2.73	4	921	1.14	4	965	0.88	3	450	0.31	4	211	3.09	3	574	1.39
04-5	6	4	483	0.95	4	483	0.95	4	259	0.66	3	4544	1.11	3	850	2.11	3	850	2.11
04-6	6	5	35757	10.46	4	779	0.56	4	1084	0.69	3	11610	2.44	5	4671	2.92	3	1834	1.43
04-7	5	4	40	2.41	3	99	0.58	4	826	0.68	3	424	0.31	3	163	0.78	3	163	0.78
04-8	5	5	35	2.17	3	102	0.52	4	140	0.42	3	573	0.24	3	111	0.60	3	111	0.60
04-9	4	4	154	2.10	2	1043	1.27	3	1942	1.26	2	996	0.67	4	35	2.21	2	1050	1.66
05-0	5	5	59	3.64	3	163	0.86	4	161	0.76	3	483	0.51	3	167	1.05	3	167	1.05
05-1	6	6	20713	28.03	5	2701	2.95	5	2976	1.45	3	18878	11.36	4	1257	3.10	4	1257	3.10
05-2	7	5	43913	47.30	4	118855	86.65	5	30518	6.14				3	158640	178.66	3	158640	178.66
05-3	7	6	1761	4.18	4	27159	24.88	5	3177	1.68	3	41447	13.08	6	513	3.09	4	13622	16.72
05-4	6	5	97355	60.37	4	989	1.63	5	3665	1.48	4	3433	1.29	5	13243	12.38	4	582	1.36
05-5	6	5	161716	88.15	5	198	0.61	5	1098	0.82	3	9550	4.61	4	347	1.05	4	347	1.05
05-6	7	6	106	1.68	4	6033	11.16	5	1827	1.23	3	49873	16.17	6	983	3.92	4	10325	16.63
05-7	6	5	184	2.67	4	944	1.92	5	13503	3.93	3	17562	9.03	4	2107	4.10	4	2107	4.10
05-8	7	6	867	4.07	5	1190	2.43	5	12285	3.82	3	61539	20.22	4	2709	7.24	4	2709	7.24
05-9	6	5	391	2.84	4	1537	2.24	4	7775	3.61	3	15829	6.85	3	2717	5.45	3	2717	5.45
06-2	6	4	143169	147.53	4	888	3.29	4	6206	5.21	3	26986	22.47	4	1709	6.91	4	1709	6.91
06-4	8				6	11535	20.81	6	30065	14.89				5	56273	131.69	5	56273	131.69
06-5	8	6	131030	233.14				6	23158	11.43									
06-6	8				5	15589	46.68	6	15181	6.67				6	76091	169.55	5	41764	133.76
06-7	9	7	20661	46.98															
06-8	8	7	203585	315.38				6	105570	33.94				7	120794	236.50			
07-0	7	6	183	6.03	5	2489	9.10	6	65866	36.77				5	6995	25.49	5	6995	25.49
07-2	10	9	16870	49.42															
07-3	10	9	9139	41.77				7	45740	37.56									
07-7	8	6	174509	306.33	5	10726	41.01	6	14420	10.47				5	38251	154.49	5	38251	154.49
07-9	8	7	53334	114.40	6	6829	19.20	6	4421	5.62				5	30148	109.49	5	30148	109.49

Table B.25: Similar to Table B.20 for the non-IPC SCHEDULE-STRIPS domain.