Optimal Additive Composition of Abstraction-based Admissible Heuristics

Michael Katz and **Carmel Domshlak***
Faculty of Industrial Engineering & Management
Technion, Israel

Abstract

We describe a procedure that takes a classical planning task, a forward-search state, and a set of abstraction-based 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 known to us abstraction-based heuristics such as PDBs, constrained PDBs, merge-and-shrink abstractions, fork-decomposition structural patterns, and structural patterns based on tractable constraint optimization.

Introduction

Admissible heuristics are critical for effective planning when either optimal or approximately-optimal solutions are required. As automated planning is known to be NP-hard even for extremely conservative problem formalisms (Bylander 1994), no heuristic should be expected to work well in all planning tasks. Moreover, even for a fixed planning task, typically no tractable heuristic will home in on all the "combinatorics" of the task in hand. The promise, however, is that (i) different heuristics will target different bolts of the planning complexity, and (ii) composing the individual strengths of numerous heuristics could allow us both solving a larger range of planning tasks, as well as solving each individual task more efficiently.

Since the late 90's, numerous (though not many) admissible heuristics for domain-independent planning have been suggested and found useful, and research in this direction becomes more and more active. In this paper we focus on the old question of how one should better orchestrate a set of admissible heuristics in the effort of 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 orchestrating a set of admissible heuristics corresponds to carefully separating the information used by the different heuristics in the set so that they values could be summed up instead of maximized over. This direction was first exploited in the works on additive pattern database (APDB) heuristics (Edelkamp 2001; Felner, Korf, & Hanan 2004), and more recently it was applied in the scope of constrained PDBs, m-reachability, and structural patterns heuristics (Haslum, Bonet, & Geffner 2005; Katz & Domshlak 2008b). The basic idea underlying all these "additive heuristic ensembles" is elegantly simple: for each problem's action a, if it can possibly be counted by more than one heuristic in the ensemble, then one should ensure that the cumulative accounting for the cost of a does not exceed its true cost in the original problem.

Until very recently, such "action-cost partitioning" was achieved in one certain manner by accounting for the whole cost of each action in computing a single heuristic, while ignoring the cost of that action (by setting it to zero) in computing all the other heuristics in the set (Edelkamp 2001; Felner, Korf, & Hanan 2004; Haslum, Bonet, & Geffner 2005). Recently, this "all-in-one/nothing-in-rest" action-cost partitioning has been generalized by Katz and Domshlak (2008b) and Yang *et al.* (2007) to *arbitrary* partitioning of the action cost among the ensembles' heuristics.

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

1. The space of alternative choices here is verbally infinite as the cost of each action can be partitioned into an arbitrary set of non-negative real numbers, sum of which does not exceed the cost of that action.
2. At least in domain-independent planning, this decision process should be fully unsupervised.
3. The last but not least, the relative quality of each action-cost partition (in terms of the informativeness 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 issues may explain why all previous works on (both domain-dependent and independent) additive heuris-

*The work of both authors is partly supported by Israel Science Foundation and C. Wellner Research Fund.

tic ensembles adopt this or another ad hoc (and search-state independent) choice of action-cost partitioning. As the result, 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 heuristics' combination was dominating the max-combination, while for the other states the opposite was the case. In the context of domain-specific APDBs, 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."

The contribution of this paper is precisely in addressing the problem of choosing the right action-cost partitioning over a given set of heuristics. Specifically, 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 admissible heuristics based on over-approximating abstractions of Π, derives an *optimal action-cost partitioning for* s (that is, a partitioning 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.
- Show that the time complexity of our procedure is *polynomial* for arbitrary sets of *all* known to us abstraction-based heuristic functions. In particular, such "procedure-friendly" heuristics include PDBs (Edelkamp 2001; Yang, Culberson, & Holte 2007), constrained PDBs (Haslum, Bonet, & Geffner 2005), merge-and-shrink abstractions (Helmert, Haslum, & Hoffmann 2007), fork-decomposition structural patterns (Katz & Domshlak 2008b), and structural patterns based on tractable constraint optimization (Katz & Domshlak 2008a).

Notice that, in particular, the estimate provided by a max-based ensemble corresponds to the estimate provided by the respective additive ensemble under *some* action-cost partitioning. As such, the max-estimate cannot exceed the one provided by the optimal action-cost partitioning, and thus, at least for the abstraction-based heuristics, we answer the aforementioned question of "to add or not to add".

Background

We consider the standard setting of cost-optimal classical planning for problems described using the SAS$^+$ representation language (Bäckström & Nebel 1995). A **SAS$^+$ planning task** is a quintuple $\Pi = \langle V, A, I, G, cost\rangle$, where

- $V = \{v_1, \ldots, v_n\}$ is a set of *state variables*, each associated with a finite domain $dom(v_i)$; each complete assignment s to V is called a *state*. I is an *initial state*, and the *goal* G is a partial assignment to V.
- A is a finite set of *actions*, where each action a is a pair $\langle \mathsf{pre}(a), \mathsf{eff}(a)\rangle$ of partial assignments to V called *preconditions* and *effects* of a, respectively, and $cost : A \to \mathbb{R}^{0+}$ is a non-negative real-valued *action cost function*.

The semantics of a planning task Π is given by its induced state-transition model, often also called *transition graph*. Searching in this transition graph corresponds to forward state-space search. In what follows we distinguish between the actual edge-weighted transition graph, and its weights-omitted, qualitative skeleton which we call *transition-graph structure*. Informally, transition-graph structures capture the combinatorics of the classical planning problems, while transition graphs annotate this combinatorics with "performance measures".

- 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*, $Tr \subseteq S \times L \times S$ is a set of (labeled) *transitions*, $s^I \in S$ is an *initial state*, and $S^G \subseteq S$ is a set of *goal states*.
- 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 $\mathcal{T}$, the **distance** $dist(s, S')$ in $\langle \mathcal{T}, \varpi\rangle$ is the cost of a cheapest (with respect to ϖ) path from s to a state in S' along the transitions of $\mathcal{T}$. Any path from s^I to S^G is a **plan** for $\langle \mathcal{T}, \varpi\rangle$, and cheapest such paths are called **optimal plans**.

The states of the TG-structure $\mathcal{T}(\Pi)$ induced by a SAS$^+$ 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 $(s, a, s') \in Tr$ iff (i) $s[v] = \mathsf{pre}(a)[v]$ whenever $\mathsf{pre}(a)[v]$ is specified and (ii) $s'[v] = \mathsf{eff}(a)[v]$ if $\mathsf{eff}(a)[v]$ is specified, and otherwise $s'[v] = s[v]$. The actual transition graph induced by Π is $\langle \mathcal{T}(\Pi), cost\rangle$.

Additive Admissible Heuristics

Our focus here is on additive ensembles of admissible heuristics, or simply, *additive heuristics*. Very often, each individual admissible heuristic for domain-independent planning is defined as the optimal cost of achieving the goals in an over-approximating *abstraction* of the planning problem in hand[1] (Pearl 1984). Such an abstraction is obtained by relaxing some constraints present in the problem, and the desire is to obtain a tractable (that is, solvable in polynomial time), yet informative abstract problem. In turn, by *additive abstraction* we refer to a set of abstractions, inter-constrained by a requirement to jointly not over-estimate the transition path costs of the original problem.

Definition 1 *An* **additive abstraction** *of a transition graph $\langle \mathcal{T}, \varpi\rangle$ is a set of pairs $\mathcal{A} = \{\langle\langle \mathcal{T}_i, \varpi_i\rangle, \alpha_i\rangle\}_{i=1}^k$ where, for $1 \le i \le k$, $\langle \mathcal{T}_i, \varpi_i\rangle$ 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, *$\alpha_i(s^I) = s_i^I$, $\alpha_i(s) \in S_i^G$ for all $s \in S^G$, and, for each pair of states $s, s' \in S$, holds*

$$\sum_{i=1}^{k} dist(\alpha_i(s), \alpha_i(s')) \le dist(s, s'). \qquad (1)$$

For $k = 1$, Definition 1 formalizes standard, non-additive abstractions, while for $k \ge 1$ it poses only a general requirement of not overestimating the distances. For our objective

[1] Admissible heuristics may also correspond to problem reformulations that are not abstractions; see our discussion later on.

of dealing with action-cost partitioning, we need a tighter binding between the original and abstract TG-structures. Specifically, we need to (i) associate each abstract transition label with a single original transition label, and (ii) verify that each original transition corresponds to an appropriate set of abstract transition *paths*.

Definition 2 *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, and $\beta_i : L_i \to L$ is an* action-associating mapping,
- *$\alpha_i(s^I) = s_i^I$, and $\alpha_i(s) \in S_i^G$ for all $s \in S^G$,*
- *for each transition $\langle s, l, s' \rangle \in Tr$, there is a path ρ from $\alpha_i(s)$ to $\alpha_i(s')$ in $\mathcal{T}_i$ such that (i) all the transitions on ρ have different labels, and (ii) for each label l' along ρ, holds $\beta_i(l') = l$.*

The notion of ABS-ensembles allows us generalizing the qualitative skeletons of various additive abstractions that are based on action-cost partitioning. In particular,

- The setting of all β_i being bijective mappings captures additive *homomorphism abstractions* such as (standard and constrained) APDBs (Yang, Culberson, & Holte 2007; Haslum, Bonet, & Geffner 2005), as well as (additive) merge-and-shrink abstractions (Helmert, Haslum, & Hoffmann 2007).
- The (possibly confusing at first view) setting of all α_i being injective (with, possibly, $|S| < |S_i|$) 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), etc.
- Each individual abstraction in an ABS-ensemble may correspond to a *hybrid (homomorphism/embedding) abstraction* such as these induced by some structural patterns (Katz & Domshlak 2008b).
- Importantly, 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 1 relates ABS-ensembles and additive abstractions induced by the former via action-cost partitioning (expressed by Eq. 2). Worth noting here that the generality of Definition 2 and Theorem 1 is not for an exercise only—later we exploit it to establish some computational results of an adequate generality. (Due to the lack of space, the proofs are given in the full TR.)

Theorem 1 (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*

$$\forall l \in L : \sum_{i=1}^{k} \sum_{l' \in \beta_i^{-1}(l)} \varpi_i(l') \leq \varpi(l), \qquad (2)$$

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

The proof of Theorem 1 is completely technical by connecting Definitions 1 and 2. In what follows, if $\Pi = \langle V, A, I, G, cost \rangle$ is a SAS$^+$ planning task, $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^k$ is an ABS-ensemble of $\mathcal{T}(\Pi)$, and $\{\varpi_i : L_i \to \mathbb{R}^{0+}\}_{i=1}^k$ is a set of transition cost functions, we say that $\mathcal{A} = \{\langle \langle \mathcal{T}_i, \varpi_i \rangle, \alpha_i \rangle\}_{i=1}^k$ *is an additive abstraction of* Π *with respect to* $\mathcal{AE}$, denoted by $\mathcal{A} \in_\Pi \mathcal{AE}$, if

$$\forall a \in A : \sum_{i=1}^{k} \sum_{l \in \beta_i^{-1}(a)} \varpi_i(l) \leq cost(a) \qquad (3)$$

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 3 *Let $\Pi = \langle V, A, I, G, cost \rangle$ be a SAS$^+$ planning task with state set S, and $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^k$ 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 $\sum_{i=1}^k dist(\alpha_i(s), S_i^G)$. 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 3 specifies the set of *all* additive heuristics for Π obtainable via action-cost partitioning over a given ABS-ensemble. Most importantly, it also specifies an *upper-bound* on the heuristic estimate $h_\mathcal{AE}(s)$ that can possibly be obtained for a state s on the basis of that (infinite) set of additive heuristics. The admissibility of each heuristic $h_\mathcal{A}$ in that space is immediate from Definition 1, and thus the proof of Theorem 2 is straightforward.

Theorem 2 (Optimal Admissibility) *For any SAS$^+$ planning task $\Pi = \langle V, A, I, G, cost \rangle$, any ABS-ensemble $\mathcal{AE}$ of $\mathcal{T}(\Pi)$, and any state s of Π, we have $h_\mathcal{AE}(s) \leq dist(s, S_G)$.*

LP-Optimization

Having specified the notion of optimal additive heuristic $h_\mathcal{AE}$, we now proceed with the computational part of the story. Suppose we are given a SAS$^+$ planning task $\Pi = \langle V, A, I, G, cost \rangle$ with state set 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 individual additive heuristics $h_\mathcal{A}$ 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 big question now is whether $h_\mathcal{AE}$ can be efficiently computed. Here we characterize a family of ABS-ensembles for which the answer to this question is affirmative, and show that this family comprises all the abstraction-based heuristic schemes suggested so far for domain-independent classical planning.

Our characterization is constructive in terms of *computability of $h_\mathcal{AE}(s)$ via a compact*[2] *linear program induced*

[2] For the sake of readability, we use "compact" as a synonym of "being of size $O(poly(|\Pi|))$".

by the triplet of Π, $\mathcal{AE}$, *and* s. We begin with introducing a set of linear constraints specifying all possible cost partitions of the actions $a \in A$ over their representatives $\cup_{i=1}^{k} \beta_i^{-1}(a)$ in the components of $\mathcal{AE}$. For each (abstract) label $l \in \cup_{i=1}^{k} L_i$, let w_l be a non-negative real-valued variable uniquely associated with l, and let the set of all these "label-cost" variables being denoted by $\overrightarrow{w}$. The (linear) *additivity constraint* $\mathcal{C}^{\mathsf{add}}(\overrightarrow{w})$ *of* Π *on* $\mathcal{AE}$ mirrors Eq. 3 as

$$\forall a \in A: \sum_{i=1}^{k} \sum_{l \in \beta_i^{-1}(a)} w_l \leq cost(a), \tag{4}$$

with $\mathscr{H}^{\mathsf{add}}$ being the convex polyhedron specified by $\mathcal{C}^{\mathsf{add}}(\overrightarrow{w})$. 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 notion of additivity constraint $\mathcal{C}^{\mathsf{add}}(\overrightarrow{w})$, we now proceed with characterizing our "$h_{\mathcal{AE}}$-friendly" family of *LP-optimizable* ABS-ensembles.

Definition 4 *Let* $\Pi = \langle V, A, I, G, cost\rangle$ *be a* SAS$^+$ *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* $\mathcal{C}^{\mathsf{add}}(\overrightarrow{w})$ *be the additivity constraint of* Π *on* $\mathcal{AE}$.

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

$$\forall \mathbf{w} \in \mathscr{H}^{\mathsf{add}}: \max_{\mathbf{xw} \in \mathscr{H}^{\mathcal{AE}}} f(\mathbf{x}) = h_{\mathcal{A}_{\mathbf{w}}}(s), \tag{5}$$

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

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

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

Theorem 3 (Tractability) *Given a* SAS$^+$ *planning task* Π*, and an ABS-ensemble* $\mathcal{AE}$ *of* $\mathcal{T}(\Pi)$*, if* $\mathcal{AE}$ *is LP-optimizable, then* $h_{\mathcal{AE}}(s)$ *is poly-time computable for every state* $s \in S$.

Theorem 4 (Composition) *For any* SAS$^+$ *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 "composite" ABS-ensemble* $\mathcal{AE} = \cup_{i=1}^{m} \mathcal{AE}_i$ *of* $\mathcal{T}(\Pi)$ *is also LP-optimizable.*

With these two gratifying properties of LP-optimizable ABS-ensembles in hand, in what follows we check whether any of the known families of abstraction-based heuristics actually leads to such LP-optimizable ABS-ensembles. The answer to this question turns out to be positive to a surprising extent.

Explicit Homomorphism Abstractions

Probably the most well-known family of additive abstractions corresponds to *additive pattern databases (APDBs)*. The idea behind various PDB heuristics is elegantly simple, and goes back to the seminal paper of Culberson and Schaeffer (1998). For any SAS$^+$ planning task $\Pi = \langle V, A, I, G, cost\rangle$, any subset of variables $V' \subseteq V$ defines an over-approximating *projection abstraction* $\Pi^{[V']} = \langle V', A^{[V']}, I^{[V']}, G^{[V']}, cost'\rangle$ of Π by intersecting the initial state, the goal, and all the actions' preconditions and effects with V' (Edelkamp 2001). In terms of ABS-ensembles, this can be formalized as follows.

Definition 5 *Given a* SAS$^+$ *planning task* $\Pi = \langle V, A, I, G, cost\rangle$*, and a set of variable subsets* $\{V_1, \ldots, V_k\}$ *of* V*, the* **pattern-database ABS-ensemble** *of* $\mathcal{T}(\Pi)$ *is* $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i\rangle\}_{i=1}^{k}$ *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 with* $S_i = dom(V_i)$*,* $L_i = A^{[V_i]}$*,* $s_i^I = I^{[V_i]}$*,* $S_i^G = \{s^{[V_i]} \mid s \in S^G\}$*, and, overall,* $|\mathcal{T}_i| = O(poly(|\Pi|))$*,*
- $\alpha_i(s) = s^{[V_i]}$*, and* $\beta_i(a^{[V_i]}) = a$*,*
- $\langle \alpha_i(s), l, \alpha_i(s')\rangle \in Tr_i$ *iff* $\langle s, l, s'\rangle \in Tr$.

While any additive PDB 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, the actual choice of abstraction (that is, the actual action-cost partitioning strategy) has remained an open issue (Yang, Culberson, & Holte 2007). This is exactly where LP-optimization gradually comes into the picture.

Without loss of generality, let the number of patterns k be $O(poly(\Pi))$. Computing $h_{\mathcal{A}} = \sum_{i=1}^{k} dist(\alpha_i(s), S_i^G)$ for a fixed APDB 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 $dist(\alpha_i(s), S_i^G)$ using the Dijkstra algorithm over the *explicitly constructed* transition graph $\langle \mathcal{T}_i, \varpi_i\rangle$. However, the corresponding single-source shortest path problem also has an elegant LP formulation. Given a directed graph $G = (N, E)$ 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

$$\begin{aligned} &\max_{\overrightarrow{d}} \sum_{v'} d(v') \\ &\text{s.t. } d(v) = 0 \\ &\quad d(v'') \leq d(v') + w(v', v''), \ \ \forall (v', v'') \in E \end{aligned} \tag{6}$$

induces a solution to the single-source shortest path problem over G and source v.

Given that, a compact LP-encoding of a pattern-database ABS-ensemble $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i\rangle\}_{i=1}^{k}$ is obtained by (i) putting together the linear constraints as in Eq. 6 for TG-structures $\mathcal{T}_i$, (ii) replacing the edge-weight constants $w(v', v'')$ by variables associated with the corresponding transition labels, and (iii) connecting the latter label-cost variables with the proper additivity constraint. Specifically, given a SAS$^+$ planning task $\Pi = \langle V, A, I, G, cost\rangle$, and a

pattern-database ABS-ensemble $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^{k}$ of $\mathcal{T}(\Pi)$, the LP-encoding construction is as follows.

First, let the label-cost variables $\overrightarrow{w}$ contain a variable $w_{a'}$ for every abstract action $a' \in \cup_{i=1}^{m} A^{[V_i]}$; the additivity constraints are defined in terms of these label-cost variables exactly as in Eq. 4. Now, given a state s of Π, we specify the LP-encoding $\mathcal{L}(s) = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathcal{C}^{\mathcal{AE}}(\overrightarrow{xw}) \rangle$ of $\mathcal{AE}$ with respect to s as

$$\overrightarrow{x} = \bigcup_{i=1}^{k} \{d(s') \mid s' \in S_i\} \cup \{d(G_i)\}$$

$$\mathcal{C}^{\mathcal{AE}} = \begin{cases} d(s') \leq d(s'') + w_{a'}, & \forall \langle s'', a', s' \rangle \in Tr_i \\ d(s') = 0, & s' = s^{[V_i]} \\ d(G_i) \leq d(s'), & s' \in S_i^G \end{cases}, \forall i$$

$$f(\overrightarrow{x}) = \sum_{i=1}^{k} d(G_i)$$

Since each TG-structure $\mathcal{T}_i$ in a pattern-database ABS-ensemble $\mathcal{AE}$ is of size $O(poly(|\Pi|))$, it is immediate that the above LP-encoding of $\mathcal{AE}$ is both compact and poly-time constructible for any state s of Π. Hence, we have $h_{\mathcal{AE}}(s)$ being poly-time computable for any planning task Π, any pattern-database ABS-ensemble $\mathcal{AE}$, and any state s of Π. Moreover, the same single-source shortest-path problems on *explicit* transition graphs underlie heuristics corresponding to additional powerful homomorphism abstractions such as constrained PDBs (Haslum, Bonet, & Geffner 2005), and merge-and-shrink abstractions (Helmert, Haslum, & Hoffmann 2007). Hence, using the "composition" Theorem 4, we can summarize here with a tractability claim that covers arbitrary combinations of such abstractions.

Theorem 5 *Given a* SAS$^+$ *planning task* Π*, and an ABS-ensemble* $\mathcal{AE} = \{\langle \mathcal{T}_i, \alpha_i, \beta_i \rangle\}_{i=1}^{k}$ *of* $\mathcal{T}(\Pi)$*, if* $\sum_{i=1}^{k} |\mathcal{T}_i| = O(poly(|\Pi|))$*, then* $\mathcal{AE}$ *is LP-optimizable, and thus* $h_{\mathcal{AE}}(s)$ *is poly-time computable for every state* $s \in S$*.*

Hybrid Abstractions: Fork-Decomposition Structural Patterns

PDB heuristics and their enhancements are successfully exploited these days in the planning systems (Haslum, Bonet, & Geffner 2005; Haslum *et al.* 2007). However, since the reachability analysis in $\Pi^{[V_i]}$ is done by exhaustive search, each pattern of a PDB heuristic (that is, each selected variable subset V_i) is required to be as small as $O(\log |V|)$ if $|dom(v)| = O(1)$ for each $v \in V_i$, and as small as $O(1)$, otherwise. For many planning domains this constraint implies an inherent scalability limitation of the PDB heuristics. One attack on this limitation *within* the scope of homomorphism abstractions correspond to the (already covered by Theorem 5) merge-and-shrink abstractions that sophisticatedly compress the abstract transition graphs to keep them within the reach of exhaustive graph searching (Helmert, Haslum, & Hoffmann 2007). Another recent proposal generalizes PDB abstractions to what is called *structural patterns* (Katz & Domshlak 2008b).

The basic idea behind structural patterns is in abstracting the problem in hand into instances of provably tractable fragments of optimal planning, alleviating by that the limitation of PDBs to use projections of only low dimensionality. In particular, Katz and Domshlak (2008b) specify a family of *causal-graph structural patterns* (CGSPs), and introduce a concrete instance of CGSPs called *fork-decomposition*. In a one-paragraph summary, the mechanism of fork-decomposition works as follows.

The causal graph $CG(\Pi) = (V, E)$ of a SAS$^+$ planning task $\Pi = \langle V, A, I, G, cost \rangle$ with variables $V = \{v_1, \ldots, v_n\}$ is a digraph over nodes V. An arc (v, v') belongs to $CG(\Pi)$ iff $v \neq v'$ and there exists an action $a \in A$ such that $\mathsf{eff}(a)[v']$, and either $\mathsf{pre}(a)[v]$ or $\mathsf{eff}(a)[v]$ are specified. For each variable $v_i \in V$, let $V_i^{\mathsf{f}} \subseteq V$ contain v_i and all its immediate successors in $CG(\Pi)$, and $V_i^{\mathsf{i}} \subseteq V$ contain v_i and all its immediate predecessors in $CG(\Pi)$. The fork-decomposition of Π is obtained by

(1) schematically constructing a set of projection homomorphism abstractions $\mathbf{\Pi} = \{\Pi^{[V_i^{\mathsf{f}}]}, \Pi^{[V_i^{\mathsf{i}}]}\}_{i=1}^{n}$ (with, possibly, each $|V_i^{\mathsf{f}}|$ and each $|V_i^{\mathsf{i}}|$ being $\Theta(n)$),

(2) reformulating the actions of the abtractions to single-effect actions only so that the causal graphs of $\Pi^{[V_i^{\mathsf{f}}]}$ and $\Pi^{[V_i^{\mathsf{i}}]}$ become "forks" and "inverted forks", respectively, rooted in v_i; after this action reformulation, the individual abstractions may cease being purely homomorphic,

(3) within each $\Pi^{[V_i^{\mathsf{f}}]}$, abstracting the domain of v_i to $\{0, 1\}$, and within each $\Pi^{[V_i^{\mathsf{i}}]}$, abstracting the domain of v_i to $\{0, 1, \ldots, k\}$ with $k = O(1)$.

This decomposition of Π provides us with the **fork-decomposition ABS-ensemble** $\mathcal{AE} = \{\langle \mathcal{T}(\Pi^{[V_i^{\mathsf{f}}]}), \alpha_i^{\mathsf{f}}, \beta_i^{\mathsf{f}} \rangle, \langle \mathcal{T}(\Pi^{[V_i^{\mathsf{i}}]}), \alpha_i^{\mathsf{i}}, \beta_i^{\mathsf{i}} \rangle\}_{i=1}^{n}$ of $\mathcal{T}(\Pi)$, with the abstraction mappings $\alpha_i^{\mathsf{f}}, \alpha_i^{\mathsf{i}}$ and the action associations $\beta_i^{\mathsf{f}}, \beta_i^{\mathsf{i}}$ 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 quite informative heuristic estimates, and (ii) each abstract problem in $\mathbf{\Pi} = \{\Pi^{[V_i^{\mathsf{f}}]}, \Pi^{[V_i^{\mathsf{i}}]}\}_{i=1}^{n}$ can be solved in polynomial time by special-purpose algorithms for the corresponding fragments of SAS$^+$ (Katz & Domshlak 2008b). However, here as well, the choice of the actual abstraction with respect to $\mathcal{AE}$ is important, and optimizing this choice is clearly of interest. Interestingly, LP-optimization can come to the rescue here as well, and this despite the TG-structures of $\mathcal{AE}$ *not* being searchable explicitly in polynomial time.

Theorem 6 *For any* SAS$^+$ *planning task* Π *over* n *variables, the fork-decomposition ABS-ensemble* $\mathcal{AE} = \{\langle \mathcal{T}(\Pi^{[V_i^{\mathsf{f}}]}), \alpha_i^{\mathsf{f}}, \beta_i^{\mathsf{f}} \rangle, \langle \mathcal{T}(\Pi^{[V_i^{\mathsf{i}}]}), \alpha_i^{\mathsf{i}}, \beta_i^{\mathsf{i}} \rangle\}_{i=1}^{n}$ *of* $\mathcal{T}(\Pi)$ *is LP-optimizable, and thus* $h_{\mathcal{AE}}(s)$ *is poly-time computable for every state* $s \in S$*.*

Our LP-encoding of a fork-decomposition ABS-ensemble $\mathcal{AE}$ corresponds to LP reformulations of the algorithms of Katz and Domshlak (2008b) for fork and inverted-fork problems with properly bounded root domains. Below we de-

scribe such an LP-encoding for an ABS-ensemble consisting of a single fork-structured abstraction with a binary root domain.

Given a SAS$^+$ planning task $\Pi = \langle V, A, I, G, cost\rangle$, let $\mathcal{AE} = \{\langle \mathcal{T}(\Pi^{\mathsf{f}}_r), \alpha^{\mathsf{f}}, \beta^{\mathsf{f}}\rangle\}$ be a fork-decomposition of $\mathcal{T}(\Pi)$ over a single fork rooted in $r \in V$. Considering that abstract problem, let us denote its variables by V', its actions by A', and its goal state $G^{[V']}$ by G'. We can assume that $G[v]$ is defined for all $v \in V' \setminus \{r\}$; all the goal-less leafs can be simply omitted from the fork. For coherence with the notation of Katz and Domshlak (2008b) we denote the (abstracted to binary-valued) domain of r by $dom(r) = \{0, 1\}$ such that $s[r] = 0$. Let $\sigma(r)$ be a $0/1$ sequence of length $1+\max_{v\in V'} |dom(v)|$, 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 non-empty prefixes of $\sigma(r)$ if $G[r]$ is unspecified, and otherwise, be the set of all prefixes of $\sigma(r)$ ending with $G[r]$.

First, let the label-cost variables $\overrightarrow{w}$ contain a variable $w_{a'}$ for every abstract action $a' \in A'$; the additivity constraints $\mathcal{C}^{\mathsf{add}}(\overrightarrow{w})$ are defined in terms of these label-cost variables via β^{f} as in Eq. 4. Now, given a state s of Π, we specify the LP-encoding $\mathcal{L}(s) = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathcal{C}^{\mathcal{AE}}(\overrightarrow{x}\overrightarrow{w})\rangle$ of $\mathcal{AE}$ with respect to s as follows. The variable set $\overrightarrow{x}$ of $\mathcal{L}(s)$ consists of three types of variables, notably

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

The variable h^{f} stands for the minimal cost of solving our fork-structured problem, and the objective function of $\mathcal{L}(s)$ 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 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 $\mathcal{C}^{\mathcal{AE}}$ of $\mathcal{L}(s)$ consists of the following sets of linear constraints.

(i) 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 $\mathsf{eff}(a')[r] = 0$,

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

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

(ii) For each leaf $v \in V' \setminus \{r\}$,

$$d(v, s[v], 0) = 0$$

and, for each $\vartheta, \vartheta' \in dom(v)$, and $1 \le i \le |\sigma(r)|$,

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

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 each $v \in V' \setminus \{r\}$, $\vartheta \in dom(v)$,

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

Likewise, for each v-changing action $a \in A'$, if $\mathsf{pre}(a)[r]$ is unspecified, then, for $\vartheta_r \in \{0, 1\}$,

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

and otherwise,

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

Semantics: Shortest-path constraints as in Eq. 6.

This finalizes our LP-encoding for an ABS-ensemble consisting of a single fork-structured abstraction with a binary root domain. Due to the lack of space, here we omit the details of the LP reformulation of the algorithm for inverted forks—while that algorithm differs from this for forks, the general spirit of the corresponding LP-encoding is quite similar. Finally, extending these encodings to ABS-ensembles containing multiple such forks and inverted forks (and possibly some other LP-optimizable abstractions) is guaranteed by the "composition" Theorem 4.

Structural Patterns and Tree-structured COPs

Fork-decomposition structural patterns are grounded in two specific fragments of tractable cost-optimal planning. In principle, it is possible that structural patterns based on some other such fragments also lead to LP-optimizable ABS-ensembles. Here we consider two such fragments that have been recently characterized by Katz and Domshlak (2008a). Both these fragments correspond to problems over binary-valued state variables and actions inducing a polytree[3] causal graph. In the first fragment, $\mathbf{P}_b$, all the causal graph's nodes have $O(1)$-bounded in-degree. In the second fragment, $\mathbf{P}(1)$, all actions are 1-dependent.

While the poly-time solution schemes provided by Katz and Domshlak for $\mathbf{P}_b$ and $\mathbf{P}(1)$ substantially differ one from another, they both correspond to *reductions of the planning problems to compact and tree-structured constraint optimization problems (COPs)*. This joint property of $\mathbf{P}_b$ and $\mathbf{P}(1)$ (that might also hold for some other problem fragments as well) turns out to be very helpful to our objective of joining $\mathbf{P}_b$- and $\mathbf{P}(1)$-based structural patterns 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 $r \in \mathcal{X}$, in what follows we refer to that rooted tree of COP via its set of *directed* edges $E = \{(x, x')\}$. In these terms, we have $\mathcal{F} = \{\varphi_x : dom(y) \times dom(x) \to \mathbb{R}^{0+} \mid (y, x) \in E\}$.

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). The bad news is that (similarly to what we had with the Dijkstra

[3]Polytree is a DAG with an acyclic induced undirected graph.

algorithm for solving PDBs) we cannot use this message-passing algorithm for our needs. The good news, however, is that such tree-structured COPs can also be solved via linear programming. Specifically, given such a problem $\mathsf{COP} = (\mathcal{X}, \mathcal{F})$, let

$$\overrightarrow{c} = \{h^{\mathsf{cop}}\} \cup \bigcup_{\substack{(x',x)\in E,\\ \overline{x}'\in dom(x')}} \{c(x|\overline{x}')\}$$

be a set of non-negative, real-valued variables, with the semantics of each $c(x|\overline{x}')$ being "optimal solution for the subtree rooted at x given $x' = \overline{x}'$". Then, the solution of LP

$$\begin{aligned} &\max_{\overrightarrow{c}} \ h^{\mathsf{cop}} \\ &\text{s.t. } \forall \overline{r} \in dom(r): \quad h^{\mathsf{cop}} \leq \sum_{(r,x)\in E} c(x|\overline{r}), \\ &\qquad \forall (x,y) \in E, \overline{x} \in dom(x), \overline{y} \in dom(y): \\ &\qquad\qquad c(y|\overline{x}) \leq \sum_{(y,z)\in E} c(z|\overline{y}) + \varphi_y(\overline{x},\overline{y}). \end{aligned} \tag{7}$$

induces a solution for COP.

Lemma 1 *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*

$$\min_{\overline{x}\in dom(\mathcal{X})} \sum_{\varphi\in\mathcal{F}} \varphi(\overline{x}) = \max_{\mathbf{c}\in\mathscr{H}^{\mathsf{COP}}} h^{\mathsf{cop}}$$

where $\mathscr{H}^{\mathsf{COP}}$ is the convex polyhedron specified by the linear constraints as in Eq. 7.

With Lemma 1 in hand, we now make two additional steps towards an LP-encoding of ABS-ensembles $\mathcal{AE}$ containing structural patterns reducible to tree-structured COPs. In each such *individual* structural pattern, each value $\varphi_y(\overline{x},\overline{y})$ of each functional component φ_y is somehow pre-computed from the costs of the actions. In our case, however, the costs of the actions in the abstract problems are not fixed in advanced, but should be determined by the process of LP-optimization. This is where our next steps come into the picture.

Consider a single COP-reducible cost-optimal planning problem. First, *suppose* that, for any *fixed* action-costs vector $\mathbf{w}^\dagger$, each functional-component value $\overline{\varphi} \equiv \varphi_y(\overline{x},\overline{y})$ corresponds to a solution value of some compact (canonical form) linear program

$$\begin{aligned} &\max f_{\overline{\varphi}}(\overrightarrow{z_{\overline{\varphi}}w}) \\ &\text{s.t. } \mathbf{A}_{\overline{\varphi}} \cdot \overrightarrow{z_{\overline{\varphi}}w} \leq \mathbf{b}_{\overline{\varphi}} \\ &\qquad \overrightarrow{w} \leq \mathbf{w}^\dagger \end{aligned} \tag{8}$$

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. 7 by (i) replacing the *constants* $\varphi_y(\overline{x},\overline{y})$ by the corresponding affine functions $f_{\overline{\varphi}}(\overrightarrow{z}_{\overline{\varphi}} \cup \overrightarrow{w})$, and (ii) for each $\overline{\varphi}$, adding its linear constraints as in Eq. 8. The extended program is still linear, and we still have

$$\min_{\overline{x}\in dom(\mathcal{X})} \sum_{\varphi\in\mathcal{F}} \varphi(\overline{x}) = \max_{\mathbf{czw}\in\mathscr{H}^{\mathsf{COP}}} h^{\mathsf{cop}}$$

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.

Now, given a SAS$^+$ 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 $\mathsf{COP}_{\Pi'}$ satisfying Eq. 8. The extended linear program specified above provides the basis for the LP-encoding of such ABS-ensembles. First, as before, let the label-cost variables $\overrightarrow{w}$ contain a variable $w_{a'}$ for every abstract action $a' \in A'$; the additivity constraints $\mathcal{C}^{\mathsf{add}}(\overrightarrow{w})$ are defined in terms of these label-cost variables via β as in Eq. 4. Now, given a state s of Π, we specify an LP-encoding $\mathcal{L}(s) = \langle \overrightarrow{x}, f(\overrightarrow{x}), \mathcal{C}^{\mathcal{AE}}(\overrightarrow{xw})\rangle$ of $\mathcal{AE}$ with respect to s as follows.

- The variable set $\overrightarrow{x} = \overrightarrow{cz}$ consists of the variables of Eqs. 7 and 8, and the objective of $\mathcal{L}(s)$ is $f(\overrightarrow{x}) = h^{\mathsf{cop}}$.
- The constraint $\mathcal{C}^{\mathcal{AE}}(\overrightarrow{xw})$ of $\mathcal{L}(s)$ consists of all the linear constraints from Eq. 7, as well as the constraint $\mathbf{A}_{\overline{\varphi}} \cdot \overrightarrow{z_{\overline{\varphi}}w} \leq \mathbf{b}_{\overline{\varphi}}$ from Eq. 8 for all functional-component values $\overline{\varphi}$ 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, again, guaranteed by the "composition" Theorem 4.

Theorem 7 *Given a SAS$^+$ 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 $k = O(poly(|\Pi|))$, and cost-optimal planning for each Π'_i is poly-time reducible to a compact and tree-structured constraint optimization problem satisfying Eq. 8, then $\mathcal{AE}$ is LP-optimizable, and thus $h_{\mathcal{AE}}(s)$ is poly-time computable for every state $s \in S$.*

The last but not least is, of course, the question of whether the requirement posed by Eq. 8 is any realistic with respect to the COPs induced by the abstract planning problems. Fortunately, Theorem 8 closes the story with some very good news on that matter.

Theorem 8 *Cost-optimal planning for any task Π in $\mathbf{P}_b \cup \mathbf{P}(1)$ is poly-time reducible to a compact and tree-structured constraint optimization problem satisfying Eq. 8.*

Discussion

Numerous recent works have suggested that additive ensembles of admissible heuristics is a powerful tool for heuristic-search systems. However, the action-cost partitioning parameter of such ensembles kept the "how to add (if at all)" question totally open. Here we described a procedure that closes this question for arbitrary ensembles of all known to

us abstraction-based heuristics such as PDBs, constrained PDBs, merge-and-shrink abstractions, fork-decomposition structural patterns, and structural patterns based on tractable constraint optimization. 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, construct an optimal additive composition of these heuristics with respect to the given state. Most importantly, the time complexity of our procedure is polynomial for arbitrary ensembles of all the above abstraction-based heuristics. We now outline some of the (in our opinion) more important challenges for future work.

Structure optimization. Probably the most important issue that remains almost entirely open is this of "structure optimization". While our framework optimizes the composition of a *given* set of TG-structures, ultimately we would like to move to even more parametric such ensembles, 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, what polytrees should (optimally) span the causal graph of the problem, etc. Note that the first step in this direction has already been made between the lines of this paper—an immediate corollary of Theorem 5 is that, for any forward-search state s, and any *fixed upper bound on the size of the PDBs*, one can construct in polynomial time the actual *optimal (for s) pattern-database ABS-ensemble*, and this simply by running LP-optimization over the ensemble of all possible such PDBs.

Additive m-reachability. As we mentioned, the seminal m-reachability heuristics h^m are not covered by our framework. While computing a single h^m heuristic for a fixed m is poly-time, h^m is not based on a problem abstraction (even in the very permissive sense of Definition 1)—the state-graph over which h^m is computed is an AND/OR-graph (and not an OR-graph such as transition graphs), each original problem state is mapped to a *set* of abstract states (and not to a concrete such state), 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 optimal additive h^m heuristics is still open and very much interesting.

LP-encodings for "double relaxations". 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 sounds and complete integer-valued LPs (such as the deletes-ignoring relaxation underlying h^+, or more recent action-ordering relaxation of van den Briel (2007)) appear to be quite natural such candidates. Things, however, are more complicated than that because, very roughly, (i) Definition 4 requires a very specific type of LP-encodings (satisfying Eq. 5), and (ii) none of the known to us ILP-to-LP "second-order" relaxations appear to be of that type. Hence, developing "Eq. 5 friendly" LP-relaxations for h^+, action-ordering relaxation, and other informative yet intractable planning relaxations is now definitely of interest.

Finally, we would like to address an almost immediate source of skepticism with respect to the practicality of our optimization procedure—using it requires solving a large LP at every search node, while typically such per-node computations are expected to be of *low* polynomial time. We believe, however, that the superior informativeness of the optimal additive heuristics has a clear potential to eventually overweight the cost of heuristic computation due to substantial reductions in the number of expanded search nodes. In other words, while the informal notion of "low polynomial time" changes with the progress of hardware technology, it is widely believed these days that P $\neq$ NP, and thus reducing the amount of expanded nodes is still the most important objective of the heuristic-search research in the long run.

References

Bäckström, C., and Nebel, B. 1995. Complexity results for SAS$^+$ planning. *Comp. Intell.* 11(4):625–655.

Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *AIJ* 69(1-2):165–204.

Culberson, J., and Schaeffer, J. 1998. Pattern databases. *Comp. Intell.* 14(4):318–334.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Edelkamp, S. 2001. Planning with pattern databases. In *ECP*, 13–34.

Felner, A.; Korf, R. E.; and Hanan, S. 2004. Additive pattern database heuristics. *JAIR* 22:279–318.

Haslum, P.; Botea, A.; Helmert, M.; Bonet, B.; and Koenig, S. 2007. Domain-independent construction of pattern database heuristics for cost-optimal planning. In *AAAI*, 1007–1012.

Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *AAAI*, 1163–1168.

Helmert, M.; Haslum, P.; and Hoffmann, J. 2007. Flexible abstraction heuristics for optimal sequential planning. In *ICAPS*, 176–183.

Katz, M., and Domshlak, C. 2008a. New islands of tractability of cost-optimal planning. *JAIR* 32.

Katz, M., and Domshlak, C. 2008b. Structural patterns heuristics via fork decomposition. In *ICAPS (this volume)*.

Pearl, J. 1984. *Heuristics — Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.

van den Briel, M.; Benton, J.; Kambhampati, S.; and Vossen, T. 2007. An LP-based heuristic for optimal planning. In *CP*, 651–665.

Yang, F.; Culberson, J.; and Holte, R. 2007. A general additive search abstraction. Technical Report TR07-06, CS Depart., Univ. of Alberta. (Extended abstract in *SARA-07*).