On Satisficing Planning with Admissible Heuristics

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

Abstract

Heuristic forward search is at the state of the art of sequential satisficing planning. The heuristics in use are, however, inadmissible, and thus give no guarantees on the quality of the obtained solution. Although there is no theoretical limitation in adopting admissible heuristics for satisficing planning, in practice there are several obstacles, such as lack of definition of one important feature, called helpful actions or preferred operators. In this paper we present a definition of preferred operators for the fork-decomposition abstraction heuristics and perform an extensive empirical evaluation on a range of domains from International Planning Competitions. In addition, we examine a mixed setting of using fork-decomposition heuristics with preferred operators derived from delete-relaxation based h_{FF} machinery.

Introduction

Heuristic search, either through progression in the space of world states or through regression in the space of subgoals, is a common and successful approach to classical planning. These days, most leading planning systems for cost-oriented classical planning adopt the forward search approach, as well as several enhancements such as helpful actions or preferred operators (Hoffmann & Nebel, 2001; Helmert, 2006) and deferred evaluation (Helmert, 2006). The notion of helpful actions refers to the actions that lead toward a solution of a simplified task (Hoffmann & Nebel, 2001), and thus are preferred over others. Deferred evaluation is another enhancement, allowing to evaluate the node only when it is expanded, and not when generated. Together, these enhancements are the cornerstone of several state-of-the-art planning systems (Richter & Helmert, 2009). All these systems adopt this or another inadmissible (but purportedly well-informed) heuristic, both for node evaluation and for deriving preferred operators. The reason for adopting the same heuristic for both is simple - the heuristic calculation is usually allowing for deriving preferred operators in little or no additional effort. The fact that inadmissible heuristics rule the area of satisficing planning should probably be attributed to two factors. First, the major breakthroughs in developing domain-independent admissible heuristics have been achieved only in the recent few years. Second, as the focus of these developments was on optimal planning, no mechanisms for deriving preferred operators in the scope of these admissible heuristics have been suggested.

With the recent substantial advances in admissible heuristics for cost-optimal classical planning, employing them not only in optimal but also in satisficing search became appealing. In this paper we show how to efficiently compute the set of preferred operators for fork-decomposition abstraction heuristics. We then empirically evaluate the efficiency of satisficing planning with these admissible heuristics. We adopt the deferred evaluation approach and investigate various settings of preferred operators, both from the fork-decomposition abstractions, as well as from the delete-relaxation based mechanism of h_{FF} (Hoffmann & Nebel, 2001) that is in use by most state-of-the-art satisficing planners.

Preliminaries

We consider classical planning tasks corresponding to state models with a single initial state and only deterministic actions. Specifically, we consider state models captured by the SAS$^+$ formalism (Bäckström & Nebel, 1995) with nonnegative action costs. Such 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 $\mathcal{D}(V') = \times_{v\in V'}\mathcal{D}(v)$. Each 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.

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.

An action a is applicable in a state s iff $s[v] = \mathsf{pre}(a)[v]$ for all $v \in \mathcal{V}(\mathsf{pre}(a))$. The set of all applicable in state s actions is denoted by $A(s)$. 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 cost-optimal plan from s_1 to s_2; $h^*(s) = \min_{s' \supseteq G} cost(s, s')$ is the custom notation for the cost of the optimal s-plan in Π. Finally, important roles in what follows are played by a pair of standard graphical structures induced by planning tasks.

- The *causal graph* $CG(\Pi)$ of Π is a digraph 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 $CG(\Pi)$.
- The *domain transition graph* $DTG(v, \Pi)$ of a variable $v \in V$ is an arc-labeled digraph over the nodes $\mathcal{D}(v)$ such that an arc (ϑ, ϑ') labeled with $\mathsf{pre}(a)[V \setminus \{v\}]$ and $cost(a)$ exists in the graph iff both $\mathsf{eff}(a)[v] = \vartheta'$, and either $\mathsf{pre}(a)[v] = \vartheta$ or $v \notin \mathcal{V}(\mathsf{pre}(a))$.

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 *additive implicit abstraction heuristics* (Katz & Domshlak, 2010b, 2010a), that are based on two fragments of tractable cost-optimal planning for tasks with fork and inverted-fork structured causal graphs. In general, for a planning task $\Pi = \langle V, A, I, G, cost\rangle$ over states S, an additive implicit abstraction of Π is denoted by a set of triplets $\mathcal{AE} = \{\langle \Pi_i, \alpha_i, \beta_i\rangle\}_{i=1}^m$, where $\Pi_i = \langle V_i, A_i, I_i, G_i, cost_i\rangle$ over states S_i is called an *abstract task*, $\alpha_i : S \mapsto S_i$ is the *state abstraction function*, and $\beta_i : A_i \mapsto A$ is the function connecting the abstract actions to their origin. The state transition system of Π_i is an abstraction of the state transition system of Π, and the admissibility of the additive result is obtained by emposing the *additive action-cost partitioning* constraint

$$\forall a \in A : \sum_{i=1}^{m} \sum_{a' \in \beta_i^{-1}(a)} cost_i(a') \leq cost(a). \quad (1)$$

Dominating Actions

Let $\Pi = \langle V, A, I, G, cost\rangle$ be a planning task over the states S. For each state $s \in S$, the set of *dominating actions* for s, denoted by $\mathrm{Pref}_\Pi(s)$, is the set of all actions $a \in A$ applicable in s, each starting some cost-optimal s-plan, that is,

$$\mathrm{Pref}_\Pi(s) = \{a \in A(s) \mid h^*(s) = cost(a) + h^*(s[\![a]\!])\}.$$

The notion of dominating actions complements the notion of *useless actions* (Wehrle, Kupferschmid, & Podelski, 2008); deciding whether an action is useless is in general as hard as planning itself.

Considering the family of abstraction heuristics, note that computing dominating actions for explicit abstractions (such as projection/pattern and merge-and-shrink abstractions) can straightforwardly be accomplished in time polynomial in the size of these abstractions. Next we show that the same holds for implicit fork-decomposition abstractions, though the corresponding procedures are not that straightforward.

Theorem 1 *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. For any set of states $S' \subseteq S$, the time and space complexity of computing the sets $Pref_\Pi(s)$ for all states $s \in S'$ is, respectively, $O(d^3 \cdot |V| + |A_r| + |S'| \cdot d \cdot |V|)$ and $O(d^2 \cdot |A|)$, where $d = \max_v \mathcal{D}(v)$.*

Proof: The proof is by a slight 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 7 (Tractable Forks) by Katz and Domshlak (2010b).

In what follows, for each of the two root's values $\vartheta \in \mathcal{D}(r)$, $\neg\vartheta$ denotes the opposite value $1 - \vartheta$, $\sigma(r|\vartheta)$ denotes the interchanging sequence of values from $\mathcal{D}(r)$ starting with ϑ, $\trianglerighteq[\sigma(r|\vartheta)]$ denotes the set of all valid prefixes of $\sigma(r|\vartheta)$, and DTG_v^ϑ denotes subgraph of $DTG(v, \Pi)$ obtained from the latter by removing arcs labelled with $\neg\vartheta$.

(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* $r : \vartheta_r$. In parallel, let $A_{\vartheta,\vartheta';\vartheta_r}$ be the set of all possible first actions of

such sequences. Both $\{p_{\vartheta,\vartheta';\vartheta_r}\}$ and $\{A_{\vartheta,\vartheta';\vartheta_r}\}$ 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 $v \in V \setminus \{r\}$, $\vartheta \in \mathcal{D}(v)$, $1 \leq i \leq d+1$, 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. In parallel, let $\tilde{A}_{\vartheta;i}(\vartheta_r)$ be the set of first actions of all such sequences, provided that these actions prevailed either by ϑ_r or nothing at all. The set $\{\tilde{g}_{\vartheta;i}(\vartheta_r)\}$ is given by the solution of the recursive equation

$$\tilde{g}_{\vartheta;i}(\vartheta_r) = \begin{cases} p_{\vartheta,G[v];\vartheta_r}, & i = 1 \\ \min\limits_{\vartheta'} \begin{bmatrix} p_{\vartheta,\vartheta';\vartheta_r} + \\ \tilde{g}_{\vartheta';i-1}(\neg\vartheta_r) \end{bmatrix}, & 1 < i \leq \delta_\vartheta \\ \tilde{g}_{\vartheta;i-1}(\vartheta_r), & \delta_\vartheta < i \leq d+1 \end{cases}, \quad (2)$$

which can be solved in time $O(d^3|V|)$, and then, $\tilde{A}_{\vartheta;i}(\vartheta_r)$ can be obtained recursively in time $O(d^3|V|)$ as

$$\tilde{A}_{\vartheta;i}(\vartheta_r) = \begin{cases} A_{\vartheta,G[v];\vartheta_r}, & i = 1 \\ \bigcup\limits_{\vartheta' \in \mathbf{M}_{\vartheta;i}(\vartheta_r)} A_{\vartheta,\vartheta';\vartheta_r}, & 1 < i \leq \delta_\vartheta \\ \tilde{A}_{\vartheta;i-1}(\vartheta_r), & \delta_\vartheta < i \leq d+1 \end{cases},$$

where

$$\mathbf{M}_{\vartheta;i}(\vartheta_r) = \{\vartheta' \mid \tilde{g}_{\vartheta;i}(\vartheta_r) = p_{\vartheta,\vartheta';\vartheta_r} + \tilde{g}_{\vartheta';i-1}(\neg\vartheta_r)\}.$$

Note that this equation is 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

$$h^*(s) = \min_{\sigma \in \trianglerighteq[\sigma(r|s[r])]} \left[cost(\sigma) + \sum_{v \in V \setminus \{r\}} \tilde{g}_{s[v];|\sigma|}(s[r]) \right] \quad (3)$$

where $cost(\sigma) = \sum_{i=2}^{|\sigma|} cost(a_{\sigma[i]})$, with $a_{\sigma[i]} \in A$ being some cheapest action changing the value of r from $\sigma[i-1]$ to $\sigma[i]$. Let $\mathrm{M}(s) \subseteq \trianglerighteq[\sigma(r \mid s[r])]$ denote the set of all sequences that obtain the minimum in Eq. 3. Now, if changing the root value first can be a part of some optimal plan, that is, $h^*(s) = h^*(s[\![a_{\sigma[2]}]\!]) + cost(a_{\sigma[2]})$, then the respective action is in $\mathrm{Pref}_\Pi(s)$. Note that using Eq. 2 we can rewrite this condition as $\tilde{g}_{s[v];|\sigma|}(s[r]) = \tilde{g}_{s[v];|\sigma|-1}(\neg s[r])$ for all $v \in V \setminus \{r\}$. Let

$$\mathrm{M}(\sigma, s) = \bigcap_{v \in V \setminus \{r\}} \{a_{\sigma[2]} \mid \tilde{g}_{s[v];|\sigma|}(s[r]) = \tilde{g}_{s[v];|\sigma|-1}(\neg s[r])\}$$

denote the set of all such cheapest actions. Thus, $\mathrm{Pref}_\Pi(s)$ can be computed as follows.

$$\mathrm{Pref}_\Pi(s) = \bigcup_{\sigma \in \mathbf{M}(s)} \left[\mathrm{M}(\sigma, s) \cup \bigcup_{v \in V \setminus \{r\}} \tilde{A}_{s[v];|\sigma|}(s[r]) \right]. \quad (4)$$

The only computation that has to be performed per search node, is the final minimization over $\trianglerighteq[\sigma(r|s[r])]$ in Eq. 3 and the union over $\mathrm{M}(s)$ in Eq. 4, and those are the lightest parts of the whole algorithm anyway. The major computations, notably those of $\{p_{\vartheta,\vartheta';\vartheta_r}\}$, $\{A_{\vartheta,\vartheta';\vartheta_r}\}$, $\{\tilde{g}_{\vartheta;i}(\vartheta_r)\}$, and $\{\tilde{A}_{\vartheta;i}(\vartheta_r)\}$, can be performed offline and shared between the evaluated states. The space required to store this information is $O(d^2|A|)$ as it contains only a $O(|A_v|)$ 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 $\trianglerighteq[\sigma(r|s[r])]$. ■

Theorem 2 *Let $\Pi = \langle V, A, I, G, cost \rangle$ be a planning task with an inverted fork causal graph with sink r and $|\mathcal{D}(r)| = b = O(1)$. For any set of states $S' \subseteq S$, the time and space complexity of computing the sets $Pref_\Pi(s)$ for all states $s \in S'$ is, respectively, $O(b|V||A_r|^{b-1} + d^3|V| + |S'||V||A_r|^{b-1})$ and $O(|V||A_r|^{b-1} + d^2|V|)$, respectively, where $d = \max_v \mathcal{D}(v)$.*

Proof: Similarly to the proof of Theorem 1, the proof of Theorem 2 is by a slight modification of the polynomial-time algorithm for computing $h^*(s)$ used for the proof of Theorem 8 (Tractable Inverted Forks) by Katz and Domshlak (2010b).

(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'. In parallel, let $A_{\vartheta,\vartheta'}$ be the set of all possible first actions of such sequences. Both $\{p_{\vartheta,\vartheta'}\}$ and $\{A_{\vartheta,\vartheta'}\}$ can be computed using the Floyd-Warshall algorithm on the domain transition graph of v in time $O(d^3|V|)$.

(2) 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 $a_\pi = a_1$ be the first action of that path, and let a "proxy" state s_π be

$$s_\pi[v] = \begin{cases} \vartheta_r, & v = r \\ G[v], & v \notin \bigcup_{i=1}^m \mathcal{V}(\mathsf{pre}(a_i)) \\ \mathsf{pre}(a_i)[v], & i = \mathrm{argmin}_j \{v \in \mathcal{V}(\mathsf{pre}(a_j))\} \end{cases},$$

that is, the nontrivial part of s_π captures the first val-

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

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

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

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

Storing the triplets (g_π, s_π, a_π) accomplishes the offline part of the computation.

(3) Now, given a search state s, we can compute

$$h^*(s) = \min_{\substack{\pi \text{ s.t.} \\ s_\pi[r] = s[r]}} \left[g_\pi + \sum_{v \in V \setminus \{r\}} p_{s[v], s_\pi[v]} \right], \quad (5)$$

storing the paths obtaining the minimum. Let $\mathrm{M}(s)$ be the set of such paths, and for each path π, let $\mathrm{M}(s,\pi) = \{a_\pi \mid a_\pi \in A(s)\}$ denote the set containing the first action of such path, if it is applicable in state s. Thus, $\mathrm{Pref}_\Pi(s)$ can be computed as follows.

$$\mathrm{Pref}_\Pi(s) = \bigcup_{\pi \in \mathbf{M}(s)} \left[\mathrm{M}(s,\pi) \cup \bigcup_{v \in V \setminus \{r\}} A_{s[v], s_\pi[v]} \right]. \quad (6)$$

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 and the first actions a_π) is $O(|V||A_r|^{b-1} + d^2|A|)$. The time complexity of the online computation per state via Eq. 5 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 $G[r]$. ∎

Experimental Evaluation

We have evaluated satisficing planning with fork-decomposition heuristics, using iterative lazy weighted A^* (Richter & Helmert, 2009) with weight schema 100, 10, 5, 3, 2, 1, with deferred evaluation and preferred operators. This setting was compared to two heuristic-search baselines, both using the same search scheme as ours, and employing the FF heuristic. The first baseline used no preferred operators at all. The second one used preferred operators from the FF heuristic (Hoffmann & Nebel, 2001). All the planners were run on one core of a 2.33GHz Intel Q8200 CPU with 4 GB memory, using 2 GB memory limit and 30 minute timeout. The summary of results is depicted in Table 1. The scoring method evaluates plan costs accordingly to the method used in International Planning Competition (IPC-2008).

Columns 2 and 3 depict the results for searching with the FF heuristic, without and with preferred operators. The evaluation covered three fork-decomposition heuristics, namely forks only ($h^{\mathcal{F}}$), inverted forks only ($h^{\mathcal{J}}$); and both forks and inverted forks. ($h^{\mathcal{FJ}}$). In order to overcome various issues caused by the 0-cost actions in IPC 2008 domains, in these domains for the purpose of heuristic evaluation all action costs were increased by 1.

The first evaluation was performed with no preferred operators, and the results are depicted in columns 4, 9, and 14, respectively. Then, preferred operators from these heuristics were added (columns 5, 10, and 15). Our first observation is that preferred operators from the implicit abstractions are not always helpful. In fact, for both $h^{\mathcal{F}}$ and $h^{\mathcal{FJ}}$, in most of the domains, the plans obtained without any preferred operators are shorter than with all preferred operators obtained from these abstractions. Note that the abstract operators change the value of a variable with either in-degree 0 or out-degree 0. In the latter case the goal value of the variable is always defined, while in sooner it is not always the case. Thus, some of these operators may be more helpful in guidance towards the goal than other. In order to check this hypothesis, we evaluated two additional settings, one taking only preferred operators changing the value of some variable with in-degree 0 in some abstraction (columns 6, 11, and 16), and another one taking only preferred operators changing the value of some variable with out-degree 0 in some abstraction (columns 7, 12, and 17). This setting changes the picture for both $h^{\mathcal{F}}$ and $h^{\mathcal{FJ}}$ heuristics, yet not for $h^{\mathcal{J}}$ heuristic where the picture is reverse. In any case, all these settings of preferred operators in most of the domains are outperformed by FF heuristic with its preferred operators. One possible reason for that is that the abstractions do not cover all planning task variables, and usually concentrated around those with goal values defined. Hence, the guidance of the actions obtained from these abstractions is greedy towards achieving the goals, and disregards other variables and intermediate goals.

In order to evaluate this assumption, we evaluated employing the fork-decomposition heuristics with preferred operators derived from the relaxed plans responsible for the FF heuristic values. The respective results are shown in columns 8, 13, and 18. Although the per-node evaluation obviously becomes more expensive, the

[1] 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.

domain	h_{FF}		$h^{\mathcal{F}}$					$h^{\mathcal{J}}$					$h^{\mathcal{FJ}}$				
	No Pref	All Pref	No Pref	All Pref	Up Pref	Lo Pref	FF Pref	No Pref	All Pref	Up Pref	Lo Pref	FF Pref	No Pref	All Pref	Up Pref	Lo Pref	FF Pref
blocks-aips2000	**34.81**	34.24	32.56	30.33	30.69	31.34	32.84	31.88	31.32	31.32	32.1	32.67	31.27	29.69	29.91	30.24	32.06
elevators-strips-seq-sat	27.26	**29.32**	11.2	9.87	16.94	14.32	12.74	8.33	8.58	8.58	8.28	13.57	24.29	19.74	20.62	24.81	26.43
logistics-AIPS98	22.55	**32.79**	20.52	14.3	18.53	20.08	28.28	20.15	26.96	18.77	30.88	31.35	20.24	14.18	18.7	21.89	28.49
openstacks-strips-seq-sat	**29.52**	29.27	29.03	25.62	27.94	21.74	29.07	23.55	23.99	23.99	22.73	29.13	29.08	25.02	28.19	21.1	28.96
pegsol-strips-seq-sat	**30**	29.85	29.95	28.86	29	28.77	29	29.95	29	29	29.95	29.95	29.95	28.67	28.75	29.53	29.9
woodworking-strips-seq-sat	12.43	**27.72**	5	6.89	5	6.89	13.11	5	6	6	5	15.67	5	6.97	5	7.74	13.08
logistics-aips2000	27.15	27.76	**27.96**	27.57	27.91	27.26	27.78	26.91	27.33	26.37	26.75	27.29	27.22	27.52	26.58	26.76	27.42
openstacks-adl-seq-sat	29.14	29.18	23.73	20.6	22.29	20.6	**29.22**	13.8	14.33	14.33	13.8	15	25.48	19.19	23.94	19.19	29.15
parcprinter-strips-seq-sat	14	14	12	22.63	20.73	20	22.8	13	26.95	26.95	13	23.93	13	25.97	26.97	26	**28.88**
scanalyzer-strips-seq-sat	24.38	25.15	22.53	21.35	21.81	22.91	21.65	22.36	**25.33**	**25.33**	22.25	22.28	21.43	21.63	21.7	22.98	22.47
sokoban-strips-seq-sat	26.83	26.88	23	22.53	24.98	20.62	23.93	**28.83**	27.73	27.73	**28.83**	27.96	24.75	21.94	23.96	21.89	24.91
transport-strips-seq-sat	12.16	18.29	19.44	14.4	17.44	15.42	**19.67**	8.3	8.34	8.34	8.3	8.94	13.22	11.93	12.61	10.87	17.82
	290.23	**324.45**	256.91	244.95	263.25	249.94	290.08	232.06	255.87	246.72	241.87	277.74	264.93	252.43	266.94	263.02	309.58

Table 1: A summary of the experimental results: plan cost. Per planner/domain, the cost of the best found plan is given by the sum of ratios over all tasks. Boldfaced results indicate the best performance within the corresponding domain. The last row summarizes the ratio over all domains.

domain	h_{FF}		$h^{\mathcal{F}}$					$h^{\mathcal{J}}$					$h^{\mathcal{FJ}}$				
	No Pref	All Pref	No Pref	All Pref	Up Pref	Lo Pref	FF Pref	No Pref	All Pref	Up Pref	Lo Pref	FF Pref	No Pref	All Pref	Up Pref	Lo Pref	FF Pref
blocks-aips2000	9.58	**32.52**	5.72	3.97	3.63	8.61	27.06	2.35	1.64	1.64	2.35	15.68	3.9	3.5	2.8	9.87	24.63
elevators-strips-seq-sat	2.05	**25.27**	0.59	0.13	0.29	0.76	10.4	0.04	0.08	0.08	0.04	3.08	1.55	0.49	0.24	1.42	24.63
logistics-AIPS98	2.57	**34.16**	1.31	0.9	0.68	2.31	18.3	0.75	1.6	0.51	7.33	23.28	1.03	1.75	0.5	4.68	21.4
pegsol-strips-seq-sat	15.32	**20.99**	3.62	4.1	4.39	4.05	13.16	3.54	3.77	3.77	3.54	10.75	5.27	3.63	3.77	5.13	13.72
sokoban-strips-seq-sat	**20.41**	12.78	3.24	4.27	4.28	3.78	9.89	7.14	8.03	8.03	7.14	13.89	6.61	8.08	8.1	7.25	11.28
woodworking-strips-seq-sat	2.45	**26.47**	1.36	3.15	2.22	3.97	6.12	0.95	3.46	3.46	1.07	7.9	1.39	3.14	2.27	4.87	6.21
logistics-aips2000	9.37	22.63	13.21	10.66	3.92	15.33	**24.89**	8.19	15.71	3.02	9.77	22.78	8.92	21.09	2.11	9.95	22.89
openstacks-adl-seq-sat	14.04	13.75	4.05	3.16	5.34	3.16	**29.74**	0.58	0.64	0.64	0.58	3.19	2.82	2.55	3.21	2.55	7.41
openstacks-strips-seq-sat	8.53	8.52	18.62	4.09	27.71	5	**28.31**	1.88	5.6	5.6	1.88	6.75	7.09	3.55	13.21	4.52	9.24
parcprinter-strips-seq-sat	2.11	3.59	2.93	7.41	5.29	12.31	9.49	4.06	11.93	11.93	4.06	11.88	4.63	5.4	5.3	**17.86**	13.98
scanalyzer-strips-seq-sat	6.04	11.64	2.62	3.88	3.84	11.49	16.01	2.12	6.78	6.78	2.12	14.03	2.84	4.86	6.64	8.66	**16.57**
transport-strips-seq-sat	1.34	15.17	1.03	0.74	0.99	0.78	13.55	0.52	0.57	0.57	0.52	3.22	1.29	0.96	1.25	0.89	**16.04**
	93.8	**227.48**	58.31	46.44	62.57	71.55	206.93	32.12	59.8	46.02	40.4	136.41	47.34	59	49.41	77.64	188

Table 2: A summary of the experimental results: expanded nodes. Per planner/domain, the number of expanded nodes of the first search is given by the sum of ratios over all tasks. Boldfaced results indicate the best performance within the corresponding domain. The last row summarizes the ratio over all domains.

overall results improve considerably. The approach is still outperformed by the baseline with preferred operators, but the substantial improvement suggests further exploration of this direction.

Another way to evaluate the performance of our approach is to look at the number of expanded nodes. Since the approach runs iteratively, we compared the results obtained for the first iteration only, that is lazy weighted A^* with the weight set to 100. Table 2 describes the results in terms of expanded nodes; with the columns having the same role as before. Overall, the picture here appears similar to this in Table 1. The main difference is that in pairwise comparison, when looking on the number of domains in which fork-decomposition heuristics achieve better performance, these heuristics appear to be slightly more competitive in terms of expanded nodes of the first iteration than in terms of plan cost of the last one.

Summary

We considered heuristic search for sequential satisficing planning and introduced a way of obtaining a set of operators from fork-decomposition implicit abstractions, to be used as preferred operators. We then discussed possible implications of such sets, and performed an empirical evaluation of several settings. We show that even a most conservative setting significantly improves performance comparatively to using no preferred operators at all. In addition, we show that combining heuristic evaluation from one heuristic family and preferred operators from another may improve the overall performance despite the overhead in computing two heuristic functions per search node.

References

Bäckström, C., & Nebel, B. (1995). Complexity results for SAS^+ planning. *Computational Intelligence*, *11*(4), 625–655.

Helmert, M. (2006). *Solving Planning Tasks in Theory and Practice*. Ph.D. thesis, Albert-Ludwigs University, Freiburg.

Hoffmann, J., & Nebel, B. (2001). The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, *14*, 253–302.

Katz, M., & Domshlak, C. (2010a). Optimal admissible composition of abstraction heuristics. *Artificial Intelligence*, *174*, 767–798.

Katz, M., & Domshlak, C. (2010b). Implicit abstraction heuristics. *Journal of Artificial Intelligence Research*, *39*, 51 – 126.

Richter, S., & Helmert, M. (2009). Preferred operators and deferred evaluation in satiscing planning. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*, pp. 273–280, Thessaloniki, Greece.

Wehrle, M., Kupferschmid, S., & Podelski, A. (2008). Useless actions are useful. In *In Proceedings of the 18th International Conference on Automated Planning and Scheduling (ICAPS 2008*, pp. 388–395. AAAI Press.