

Cost-Sharing Approximations for h^+

Vitaly Mirkis and Carmel Domshlak

Faculty of Industrial Engineering and Management
Technion—Israel Institute of Technology
Haifa, Israel

Abstract

Relaxations based on (either complete or partial) ignoring delete effects of the actions provide the basis for some seminal classical planning heuristics. However, the palette of the conceptual tools exploited by these heuristics remains rather limited. We study a framework for approximating the optimal cost solutions for problems with no delete effects that bridges between certain works on heuristic search for probabilistic reasoning and classical planning. In particular, this framework generalizes some previously known, as well as suggests some novel, tools for heuristic estimates for Strips planning.

Introduction

The automatic derivation of heuristic functions from problem descriptions in declarative action languages has been one of the key developments in recent planning research (McDermott 1999; Bonet & Geffner 2001; Haslum & Geffner 2000; Hoffmann & Nebel 2001; Edelkamp 2001; Nguyen, Kambhampati, & Nigenda 2002; Rintanen 2006). Such heuristic functions are used to estimate the distance from search states to their nearest goal states, and search algorithms can use these estimates to guide the search. If the heuristic function is also admissible, that is, never overestimates the true cost of reaching the nearest goal state, then certain well-known search algorithms are guaranteed to provide an optimal or approximately-optimal plan to the goal (Pearl 1984; Korf 1985).

So far, the major impact on developing heuristics for domain-independent Strips planning should probably be attributed to exploiting the complete (McDermott 1999; Bonet & Geffner 2001; Refanidis & Vlahavas 2001; Hoffmann & Nebel 2001) or selective (Haslum & Geffner 2000; Nguyen, Kambhampati, & Nigenda 2002; Haslum, Bonet, & Geffner 2005) ignoring-delete-effects relaxation of the problem. While computing the optimal cost h^+ for problems with no delete effects is NP-hard (Bylander 1994), approximations for this problem have led to introducing both quite informative heuristics h_{add} (McDermott 1999; Bonet & Geffner 2001) and h_{ff} (Hoffmann & Nebel 2001),

and the admissible heuristic h_{max} (Bonet & Geffner 2001). By now, these developments in approximating h^+ have become seminal, and a fruitful attempt to unify and generalize them has been recently made in (Rintanen 2006). The propositional-logic based framework suggested in (Rintanen 2006) provides a unified perspective on h_{max} , h_{add} , and h_{ff} , and allows for extending these three approximations to efficiently handle problems with conditional effects. However, a few questions of theoretical and practical interest with respect to the ignoring-delete-effects relaxation remain open. In particular, in this work we consider the following two open questions:

1. If optimal solutions are required, can we formulate a non-trivial admissible approximation of h^+ other than h_{max} ?
2. If near-optimal solutions are acceptable, are h_{add} and h_{ff} the most informative approximations possible for h^+ ?

Targeting these two questions, here we introduce a novel generalizing perspective on estimating h^+ . Inspired by works on cost-based abduction on weighted and/or dags (Charniak & Husain 1991; Shimony, Domshlak, & Santos 1997), the suggested *cost-sharing* framework is based on a parametric *cost propagation* over schematically constructed relaxed planning graphs (Hoffmann & Nebel 2001). We begin with introducing the framework, and show that h_{max} , h_{add} and h_{ff} can be casted as its instances. Next we consider the two aforementioned open questions with respect to approximating h^+ . We show that there is an admissible alternative to h_{max} within the cost-sharing framework, and this heuristic, h_{cs} , has been *successfully* used in probabilistic reasoning for more than a decade. However, what is good for “typical” problems of probabilistic reasoning turns out not to be so for “typical” problems of classical planning. We discuss the reasons for such a difference in h_{cs} ’s attractiveness between the two problem areas, and show how one can possibly improve the informativeness of h_{cs} . In general, however, we show that the latter improvement boils down to solving an NP-hard problem.

Next, putting the admissibility aside, we show that various instances of the framework can be more informative than h_{add} and h_{ff} with respect to the optimal solution cost. In particular, this can be obtained by propagating more complicated (than single scalars) pieces of information about the estimated cost of the sub-goals. We present and discuss one

such framework instance, h_{pmax} , and empirically demonstrate that it leads to discovering solutions of higher quality than h_{add} and h_{ff} while expanding significantly less search nodes than h_{max} .

Basic Formalism and Previous Work

The problems of *classical planning* correspond to state models with deterministic actions and complete information. While several languages for describing classical planning problems have been proposed, the most canonical is probably the Strips language (Fikes & Nilsson 1971). A planning problem in Strips is given by a tuple $\Pi = \langle P, A, I, G \rangle$ where P is a set of atoms (also called propositions), $I \subseteq P$ encodes the initial situation, $G \subseteq P$ encodes the goal situations, and A is a set of actions $a = \langle \text{pre}(a), \text{add}(a), \text{del}(a) \rangle$, where $\text{pre}(a) \subseteq P$, $\text{add}(a) \subseteq P$, and $\text{del}(a) \subseteq P$ capture the preconditions, add effects, and delete effects of the action a , respectively. The cost of a plan $\alpha \in A^*$ for Π is given by $C(s_0, \alpha) = \sum_{a \in \alpha} C(a)$, where the (possibly non-uniform) costs of all actions in A are assumed to be non-negative.

Research on using abstractions for heuristic-search Strips planning has began in the late 90's (Bonet & Geffner 2001; McDermott 1999; Refanidis & Vlahavas 2001; Hoffmann & Nebel 2001; Nguyen, Kambhampati, & Nigenda 2002), and one such abstraction has provided the foundations for numerous developments in heuristic-search planning. Specifically, the ‘‘ignoring-delete-effects’’ abstraction $\Pi|^+ = \langle P, A|^+, I, G \rangle$ of $\Pi = \langle P, A, I, G \rangle$ is obtained by removing the delete lists from the actions of Π , that is,

$$A|^+ = \{a' \mid a \in A, a' = (\text{pre}(a), \text{add}(a), \emptyset)\}.$$

This abstraction is known to be rather informative for a wide spectrum of planning problems, but computing the optimal cost $h^+(s)$ for $\Pi|^+$ is still an NP-hard problem (Bylander 1994). Thus, even if desirable, using h^+ as a heuristic estimate directly is problematic, and thus further approximation of h^+ is required.

An elegant admissible *approximation* h_{max} of h^+ has been suggested in (Bonet & Geffner 2001), and it corresponds to estimating the cost of achieving a set of atoms by the cost of achieving in $\Pi|^+$ the most costly atom in the set. Given a state s that should be assigned a heuristic value, these estimates are computed for all atoms $p \in P$ by performing incremental updates

$$g_s(p) := \min_{\substack{a \in A, \\ p \in \text{add}(a)}} \{g_s(p), C(a) + g_s(\text{pre}(a))\}, \quad (1)$$

where, for a set of atoms $P' \subseteq P$,

$$g_s(P') := \max_{p \in P'} g_s(p). \quad (2)$$

This iterative procedure starts with $g_s(p) = 0$ if $p \in s$, and $g_s(p) = \infty$, otherwise, and runs until the costs $g_s(p)$ reach the fix-point. The heuristic estimate for s is then set to $h_{\text{max}}(s) := g_s(G)$.

The max-heuristic h_{max} is admissible, but typically is ‘‘too admissible’’. The main weakness of h_{max} is that it ignores independence between achieving different subgoals,

and this leads to poor heuristic estimates in domains that exhibit a substantial degree of parallelism. To alleviate this shortcoming, one can possibly trade admissibility of heuristic for its informativeness. In some sense, one can see this move as trading the expected quality of the solution for efficiency of its generation. This direction has led to formulation of the (much more informative) heuristics h_{add} (Bonet & Geffner 2001) and h_{ff} (Hoffmann & Nebel 2001), where the former is obtained by replacing the maximization in Eq. 2 with summation, and the latter estimates $h^+(s)$ with the (computable in low polynomial time) cost of *some* plan from s in $\Pi|^+$. In what follows, we distinguish between h_{ff} and h_{FF} with the former corresponding to the principal idea of basing the estimate on some relaxed plan, and the latter corresponding to the concrete procedure of the FF planner for computing such a plan (Hoffmann & Nebel 2001).

Cost-Sharing Heuristics

We now define a computational framework that characterizes what we call the *cost-sharing* class of approximations of h^+ . All these approximations can be *schematically* specified via the same core mechanism of cost propagation over planning graphs, with the difference being in the actual propagated costs.

We begin with some useful notation. To indicate parent and child relations in a digraph (V, E) we use the convention that parents are ‘‘before’’ children along with subscript and superscript capturing the ‘‘immediately before’’ and ‘‘immediately after’’ (i.e., parents and children) relations. For convenience, this notation is used for both nodes and edges; E_v and E^v are the sets of incoming and outgoing edges of node v , U_v and U^v are the parent and children nodes of v , and v_e, v^e are the source and end nodes of edge e , respectively.

Given a state s and a goal G , the planning graph $\text{eRPG}(s, G)$ required for our purposes is very similar to the standard relaxed planning graph (Hoffmann & Nebel 2001). The procedure build-RPG for constructing $\text{eRPG}(s, G)$ is depicted in Figure 1. Similarly to its well-known relative, eRPG is a layered acyclic digraph with two kinds of nodes: fact nodes and action nodes. The layers alternate between fact layers $P_{(0)}, P_{(1)}, \dots$, and action layers $A_{(0)}, A_{(1)}, \dots$, where a pair of layers $P_{(i)}, A_{(i)}$ together make up a ‘‘time step’’.

The specifics of eRPG are in (i) the depth ub to which the graph is constructed, and (ii) the additional special action layer $A_{(ub+1)}$. The construction of $\text{eRPG}(s, G)$ ensures that the graph captures an optimal plan from s in $\Pi|^+$. For that, the depth of $\text{eRPG}(s, G)$ is controlled by an *upper bound* ub on the number of actions in the *shortest optimal plan* from s in $\Pi|^+$. While there are several alternatives for setting ub , one very simple (though typically horribly inefficient) alternative would be $ub = |P|$. However, much better alternatives are abundant. For instance, for ‘‘truly classical’’ planning problems with uniform cost actions, ub can be set to the number of actions in a *plan* from s in $\Pi|^+$ (that is, to the h_{ff} estimate (Hoffmann & Nebel 2001)), that can be computed at relatively low cost *during* the construction of

eRPG. In case of non-uniform action costs, the issue of devising an effective upper bound ub is somewhat more open. However, if the variance of the action costs is not too high, then the effectiveness of the bound

$$ub = \operatorname{argmax}_{A' \in A, C(A') \leq C} |A'|, \quad (3)$$

where C is the cost of the relaxed plan extracted by the FF procedure, should be close to the effectiveness of $ub = h_{\text{FF}}(s)$ on the problems with uniform-cost actions¹.

Once eRPG is constructed up to the layers $A_{(ub)}$ and $P_{(ub+1)}$, if some of the goal atoms are not present in $P_{(ub+1)}$, then G is proven to be unreachable from s in $\Pi|^{+}$ (and thus, in Π), and therefore we can safely set $h(s) = \infty$. Otherwise, we proceed with extending eRPG with the action layer $A_{ub+1} = \{\tilde{a}\}$, with \tilde{a} being a dummy relaxed action with $pre(\tilde{a}) = G$, $add(\tilde{a}) = \emptyset$, and $C(\tilde{a}) = 0$, and connect between the goal fact nodes in $P_{(ub+1)}$ and \tilde{a} .

Having constructed the planning graph $\text{eRPG}(s, G)$, we proceed with the cost propagation step. Here as well, we begin with providing some essential formalism. An *and-or dag* $\Delta = (V, E)$ is a connected acyclic digraph with a partition of nodes into and-nodes aV and or-nodes oV , and a single sink (out-degree 0) node $r_{\Delta} \in aV$ called the root node. Each and-or dag Δ induces a set $\mathcal{A}(\Delta)$ of its *and-dags*; An and-dag $\delta \in \mathcal{A}(\Delta)$ is obtained from Δ by removing all but one descendants from each or-node in Δ .

A *weighted and-or dag* (*waodag*) Δ_w is a pair (Δ, w) where Δ is an and-or dag, and $w : V \rightarrow \mathbb{R}^{0+}$ is a non-negative real-valued weight function from nodes. The weight of Δ_w is defined as

$$\begin{aligned} w(\Delta_w) &= \min_{\delta \in \mathcal{A}(\Delta_w)} w(\delta), \\ w(\delta) &= \sum_{v \in V_{\delta}} w(v), \end{aligned}$$

and $\mathcal{A}_{\min}(\Delta_w) = \{\delta \in \mathcal{A}(\Delta_w) \mid w(\delta) = w(\Delta_w)\}$ is the set of all *minimal and-dags* of Δ_w .

Now, consider the $\text{eRPG}(s, G)$ graph constructed by the build-RPG procedure, and let us lift our cost function C from actions to (both fact and action) nodes of eRPG as $C(a_{(i)}) = C(a)$, and $C(p_{(i)}) = 0$.

Proposition 1 *Considering $\text{eRPG}(s, G)$ as a waodag Δ_C with $oV = \bigcup P_{(i)}$, $aV = \bigcup A_{(i)}$, and root \tilde{a} , we have $C(\Delta_C) = h^+(s)$.*

The proof of Proposition 1 shows that each and-dag in $\mathcal{A}_{\min}(\Delta_C)$ corresponds to an optimal plan for $\Pi|^{+}$, and at least one optimal plan for $\Pi|^{+}$ corresponds to an and-dag in $\mathcal{A}_{\min}(\Delta_C)$. In particular, this implies that approximating h^+ is equivalent to approximating $C(\Delta_C)$, and this simple observation leads to specifying a general cost propagation scheme for approximating $h^+(s)$. The skeleton of this weight propagation is given by Eq. 4. Let Δ_C be a waodag

¹The bound provided by Eq. 3 can be computed efficiently by greedily expanding A' with actions from A in the increasing order of their cost.

```

procedure build-RPG ( $s, A|^{+}, G, ub$ ),
 $P_{(0)} = s$ 
for  $t := 0 \dots ub$  do
   $A_{(t)} := \{a_{(t)} \mid a \in A|^{+}, pre(a) \subseteq P_{(t)}\}$ ;
   $E_{A_{(t)}} := \{(p_{(t)}, a_{(t)}) \mid p \in pre(a)\}$ 
   $P_{(t+1)} := \{p_{(t+1)} \mid p \in add(a), a_{(t)} \in A_{(t)}\}$ ;
   $E_{P_{(t+1)}} := \{(a_{(t)}, p_{(t+1)}) \mid p \in add(a)\}$ 
if  $G \not\subseteq P_{(t)}$  then return FALSE
 $A_{(t+1)} := \{\tilde{a}\}$ ;  $E_{A_{(t+1)}} = \{(p_{(t+1)}, \tilde{a}) \mid p \in G\}$ 
return TRUE

```

Figure 1: Building eRPG from s to G .

with $\Delta = (V, E)$. The heuristic estimate $h(s)$ is defined via the *parametric cost estimator* $\varpi : V \cup E \rightarrow \mathcal{D}$ as

$$\begin{aligned} h(s) &= f(\varpi(\tilde{a})) \\ \varpi(v) &= \begin{cases} \varphi_a(v, E_v), & v \text{ is an action node} \\ \varphi_p(v, E_v), & v \text{ is a fact node} \end{cases} \quad (4) \\ \varpi(e) &= \begin{cases} \psi_a(v_e), & v_e \text{ is an action node} \\ \psi_p(v_e), & v_e \text{ is a fact node} \end{cases} \end{aligned}$$

where \mathcal{D} is a set, and $\Lambda = \langle \varphi_a, \varphi_p, \psi_a, \psi_p, f \rangle$ is a set of functional parameters of ϖ , with $\varphi_a, \varphi_p : V \times 2^E \rightarrow \mathcal{D}$, $\psi_a, \psi_p : V \rightarrow \mathcal{D}$, and $f : \mathcal{D} \rightarrow \mathbb{R}^{0+}$. Note that planning graphs are not really required for computing Eq 4, and the latter can be done iteratively *la* Eqs. 1-2. However, later we show that considering the framework through the lens of Proposition 1 does provide helpful insights into the process of cost propagation.

Proposition 2 *Heuristic functions h_{\max} , h_{add} , and h_{ff} can be casted as special cases of the framework (4).*

While the h_{ff} -instance of the framework is given later in the paper (in Eq. 11), h_{add} and h_{\max} instances of (4) can be formulated by setting $\mathcal{D} = \mathbb{R}^{0+}$, $f = \text{identity function}$, and

$$\begin{aligned} \varphi_p(p_{(i)}, E_{p_{(i)}}) &= \begin{cases} 0, & i = 0 \\ \min_{e \in E_{p_{(i)}}} \varpi(e), & i > 0 \end{cases}, \\ \varphi_a(a_{(i)}, E_{a_{(i)}}) &= C(a_{(i)}) + \sum_{e \in E_{a_{(i)}}} \varpi(e), \quad (5) \\ \psi_p(p_{(i)}) &= \varpi(p_{(i)}), \quad \psi_a(a_{(i)}) = \varpi(a_{(i)}) \end{aligned}$$

for h_{add} , replacing for h_{\max} the internal summation of φ_a in Eq. 5 by maximization. Informally, in both cases, the cost estimate $\varpi(u)$ for a node u is determined by the estimates for its incoming edges E_u , and the cost estimate $\varpi(e)$ for an edge e is determined given the estimate of its source node u_e . The process of cost propagation finishes by computing the cost estimate $\varpi(\tilde{a})$ for the special action node \tilde{a} , and setting the heuristic estimate to $\varpi(\tilde{a})$.

From Probabilistic Reasoning to Strips

While h_{\max} , h_{add} , and h_{ff} fit the cost-sharing framework, this in itself is not especially helpful. However, the generality of the framework may allow us specifying and studying

a wider palette of heuristic functions. In particular, the question we consider in this section is whether some non-trivial instances of Eq. 4 other than h_{\max} provide us with admissible approximations of h^+ , and how the informativeness of these approximations compares to this of h_{\max} .

Interestingly, the answer to the first question appears to be affirmative. More than a decade ago, Charniak and Husain suggested an admissible estimate for the cost of woadags, and this estimate has been successfully used in probabilistic reasoning and cost abduction (Charniak & Husain 1991). In our terms, this estimate h_{cs} is given by $\mathcal{D} = \mathbb{R}^{0+}$, $f =$ identity function, and

$$\begin{aligned} \varpi(p_{(i)}) &= \begin{cases} 0, & i = 0 \\ \min_{e \in E_{p_{(i)}}} \varpi(e), & i > 0 \end{cases}, \\ \varpi(a_{(i)}) &= C(a_{(i)}) + \sum_{e \in E_{a_{(i)}}} \varpi(e), \\ \varpi(e^{p_{(i)}}) &= \frac{\varpi(p_{(i)})}{|E^{p_{(i)}}|}, \quad \varpi(e^{a_{(i)}}) = \frac{\varpi(a_{(i)})}{|E^{a_{(i)}}|}. \end{aligned} \quad (6)$$

Note that the cost propagation to the nodes in Eq. 6 is identical to this in Eq. 5 for (inadmissible) h_{add} . The difference is in the cost propagation to the edges. Unlike for h_{add} , the cost estimate of an action/fact node in Eq. 6 is not fully propagated along each outgoing edge, but *partitioned* between them. This syntactically slight adaptation Eq. 5 is sufficient to make the heuristic admissible, and the proof of admissibility is rather straightforward from the results in (Charniak & Husain 1991).

The h_{cs} heuristic has been shown very effective on various problems of probabilistic reasoning (Charniak & Husain 1991; Shimony, Domshlak, & Santos 1997), and a priori this suggests that the same heuristic could be effective on the planning problems as well. However, the informativeness of h_{cs} in our context is more problematic. In our experience², using A* equipped with h_{cs} for planning as forward search turns out to be as (in)efficient as using uniform-cost search that completely ignores the heuristic estimates. Figure 2 illustrates that on problems from the Blocksworld and Logistics domains, but the phenomenon holds for all benchmarks from the recent planning competitions. The problem is that the values of $h_{cs}(s)$ for states encountered in searching for a plan are typically so low that the choice of the search node to expand is basically determined by the cost-so-far part $g(s)$ of the node evaluation function $f(s) = g(s) + h(s)$.

It appears that such a dramatic difference in effectiveness of h_{cs} in estimating cost of the woadags in planning and in probabilistic reasoning can be explained. The major factor affecting the informativeness of h_{cs} is the *degree of erosion* of the propagated action costs. This erosion is caused by the distribution of the cost of a node among its indirect descendants in the graph. One property of the woadags that dramatically affects the cost erosion is the *out-degree* of the nodes. While the out-degree of the nodes in woadags describing probabilistic models tend to be low, this is not the

²Our implementation is based on the A* algorithm of the HSP2 planner, and we are thankful to B. Bonet and H. Geffner for making its code publicly available.

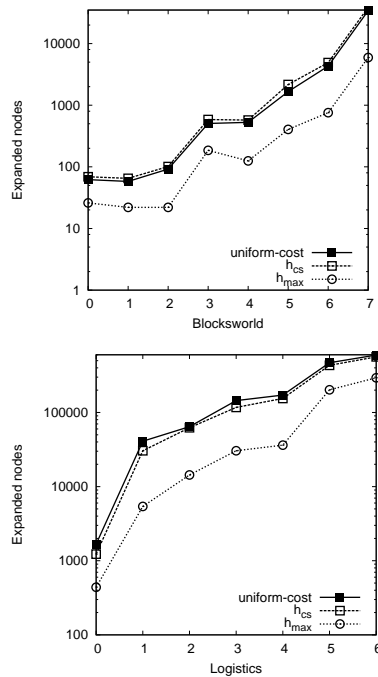


Figure 2: Number of expanded nodes on Blocksworld and Logistics problems with uniform-cost search and A* with h_{\max} , and h_{cs} heuristics.

case in planning graphs, or at least, in planning graphs coming from the standard classical planning benchmarks³. The problem is especially severe with the fact nodes, out-degrees of whose correspond to the number of applicable actions requiring these facts. Due to the combinatorial structure of the benchmarks, these numbers for facts tend to be extremely large. For instance, the out-degree of a fact node $\text{clear}(b)_{(i)}$ in Blocksworld gets as large as $2B$, where B is the total number of blocks.

In the past it was already observed that the problem of Eq. 6 with large node out-degrees can be partly alleviated (Charniak & Husain 1991; Shimony, Domshlak, & Santos 1997). Specifically, it can be shown that ψ_p and ψ_a in Eq. 6 can be replaced with

$$\varpi(e^{p_{(i)}}) = \frac{\varpi(p_{(i)})}{\kappa(p_{(i)})}, \quad \varpi(e^{a_{(i)}}) = \frac{\varpi(a_{(i)})}{\kappa(a_{(i)})} \quad (7)$$

where the *effective out-degree* $\kappa(v)$ of a node v is the size of a largest subset of E^v that belongs to some and-dag in $\mathcal{A}(\text{eRPG})$, that is, $\kappa(v) = \max_{\delta \in \mathcal{A}(\text{eRPG})} |E^v \cap \delta|$. Interestingly, comparing the modified Eq. 6 with the formulation of h_{add} as in Eq. 5 shows that h_{add} constitutes a variant of h_{cs} that *assumes* $\kappa(v) = 1$ for all nodes v . (Obviously, this assumption rarely holds.) The question, however, is whether determining the effective out-degrees of the eRPG's nodes can always be done efficiently. Unfortunately, Proposition 3 below shows that this is not the case.

³To what degree this picture is typical for real-world, structured planning problems is less clear, and we leave this question to practitioners facing concrete planning problems.

Proposition 3 Given an and-or dag $\Delta = (V, E)$, and $v \in V$, determining the effective out-degree $\kappa(v)$ in Δ is NP-hard.

The proof of Proposition 3 is by a polynomial reduction from the MAX-2-SAT problem. It is worth noting that, in our context, the and-nodes of the waodags correspond to actions, and thus their in- and out-degrees correspond to the number of preconditions and add effects, respectively. The latter parameters are typically bounded by a small constant, and our reduction from MAX-2-SAT respects this property.

The efficiency of determining the effective out-degree of the nodes of eRPG is not the only issue with h_{cs} . While knowing the latter has the potential to reduce the cost erosion, it seems unlikely that this reduction will typically change the picture significantly. First, large $\kappa(v)$ in planning problems is more of a rule than of an anomaly. For illustration, consider a Logistics problem in which all P packages and a truck t are initially located at the same location l , with the latter being the target location for none of the packages. In this case, the fact node $\text{at}(t, l)_{(0)}$ will have outgoing edges to at least the P action nodes $\text{load}(p, t, l)_{(0)}$, and all these edges will be part of a plan (and thus an and-dag) in which t takes all the packages from l to their target destinations. In fact, this plan can even be optimal.

The second problem is that the node out-degrees are not the only cause of the cost erosion in h_{cs} . Consider two nodes $v_{(t)}$ and $v'_{(t')}$ in eRPG such that $t > t'$, and assume that the cost propagation as in Eqs. 6-7 results in cost estimate $\varpi(v'_{(t')})$ contributing to the cost estimate $\varpi(v_{(t)})$. It is not hard to verify that the marginal cost contribution of $v'_{(t')}$ to $\varpi(v_{(t)})$ may decrease exponentially with the distance $t - t'$ between the two nodes, and this unless the effective out-degree of the eRPG nodes is (known to be) $= 1$. In planning benchmarks, however, $\kappa(v) = 1$ never happens for fact nodes, and very rarely for action nodes. On the other hand, even efficiently bounded depth of eRPG is typically such that most of the contributions to the node's cost are getting negligible.

Propagating Compound Cost Estimates

The question that the previous section leaves open is whether a setting of the framework (4) results in a more informative that h_{\max} and h_{cs} , admissible estimate for h^+ . One can, however, also try exploiting the generality of the cost-sharing framework to derive non-admissible heuristics lying in between $h_{\text{add}}/h_{\text{ff}}$ and h_{\max} . That is, heuristics leading to discovering solutions of higher quality than $h_{\text{add}}/h_{\text{FF}}$, but expanding less nodes than h_{\max} . Here we suggest and explore one such setting of the framework, which we refer to as h_{pmax} (short for *pairwise max*). The source of this name for the heuristic is now explained.

The core difference between h_{pmax} on one hand, and h_{\max} , h_{add} , and h_{cs} , on the other hand, is that in h_{pmax} we propagate not single costs, but vectors of costs. Specifically, each node v in eRPG is annotated with a *cost vector* $\vec{\omega}_v$ in $\mathcal{D} = \mathbb{R}^{|P|}$. The dimensions of $\mathbb{R}^{|P|}$ are associated with facts P , and $\vec{\omega}_v[p]$ stands for the component of $\vec{\omega}_v$ associ-

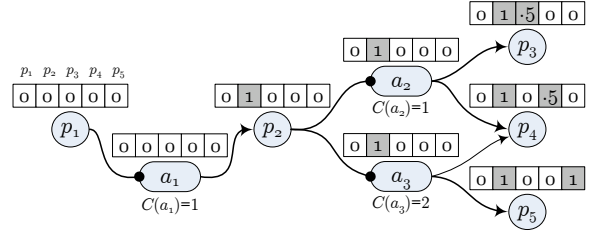


Figure 3: Example of cost vectors propagation over a prefix of a simple eRPG (noops omitted). Here we have $P = \{p_1, \dots, p_5\}$ and $A = \{a_1, a_2, a_3\}$, where $\text{pre}(a_1) = \{p_1\}$, $\text{pre}(a_2) = \text{pre}(a_3) = \{p_2\}$, and $\text{add}(a_1) = \{p_2\}$, $\text{add}(a_2) = \{p_3, p_4\}$, $\text{add}(a_3) = \{p_3, p_4\}$.

ated with $p \in P$. For ease of presentation, each action node $a_{(i)}$ is also schematically annotated with a *scalar estimate* $w_{a_{(i)}}$ derived from the cost vector $\vec{\omega}_{a_{(i)}}$.

First, for each $p_{(0)} \in P_{(0)}$, we set $\vec{\omega}_{p_{(0)}}[q] = 0$ for all $q \in P$. (See Figure 3 for an illustration by example.) The cost vectors for the rest of the nodes of eRPG are then iteratively defined (and computed) as follows. Given the cost vectors of $P_{(t)}$, for each action node $a_{(t)} \in A_{(t)}$, we set $\vec{\omega}_{a_{(t)}}$ and $w_{a_{(t)}}$ to

$$\vec{\omega}_{a_{(t)}}[q] = \max_{(p_{(t)}, a_{(t)}) \in E_{a_{(t)}}} \{\vec{\omega}_{p_{(t)}}[q]\},$$

$$w_{a_{(t)}} = \sum_{q \in P} \vec{\omega}_{a_{(t)}}[q] \quad (8)$$

Informally, $w_{a_{(t)}}$ estimates the cost of being able to apply a at time t , with the “prerequisite set” $\text{Prq}(a_{(t)}) = \{q \in P \mid \vec{\omega}_{a_{(t)}}[q] \neq 0\}$ containing the propositions that should be achieved prior to applying a at t , and the corresponding entries $\vec{\omega}_{a_{(t)}}[q]$ capturing the marginal cost of achieving the “prerequisite” q . Importantly, while Eq. 8 estimates the cost of applying a at t as the sum of achieving its prerequisites, the prerequisites are treated as independent only when it is considered to be safe: If some fact $q \in \text{Prq}(a_{(t)})$ is considered contributing to achieving more than one precondition of a at time t , then only one such contribution is taken by Eq. 8 into account.

Given the cost vectors and scalar estimates of $A_{(t)}$, for each fact node $p_{(t+1)} \in P_{(t+1)}$, we select an action node $\widehat{a}_{(t)} \in A_{(t)}$ that provides $p_{(t+1)}$ and satisfies

$$\widehat{a}_{(t)} = \underset{(a_{(t)}, p_{(t+1)}) \in E_{p_{(t+1)}}}{\text{argmin}} \left\{ w_{a_{(t)}} + \frac{C(a_{(t)})}{|E^{a_{(t)}}|} \right\}. \quad (9)$$

The cost vector of each $p_{(t+1)} \in P_{(t+1)}$ is then defined as:

$$\vec{\omega}_{p_{(t+1)}}[q] = \begin{cases} \vec{\omega}_{\widehat{a}_{(t)}}[q], & p_{(t+1)} \neq q_{(t+1)} \\ \vec{\omega}_{\widehat{a}_{(t)}}[q] + \frac{C(\widehat{a}_{(t)})}{|E^{\widehat{a}_{(t)}}|}, & p_{(t+1)} = q_{(t+1)} \end{cases}, \quad (10)$$

where $E^{\widehat{a}_{(t)}}$ is the set of edges outgoing from $\widehat{a}_{(t)}$. This way, $\vec{\omega}_{p_{(t+1)}}$ captures the estimated cost of achieving p at

time $t + 1$, and $Prq(p_{(t+1)})$ captures the corresponding set of purported prerequisites. The cost propagation finishes by computing the cost estimate $w(\tilde{a})$, and the heuristic value is set to $h_{\text{pmax}}(s) = w(\tilde{a})$. (As a practical comment, we note that the propagated cost vectors \vec{w} will typically be very sparse, and thus propagating them as sets of their non-zero entries significantly speeds-up computing h_{pmax} .)

Let us now consider h_{pmax} more closely. Aiming to avoid overestimates in cost propagation, the cost of applying an action is partitioned in Eqs. 9-10 among its add effects. h_{pmax} inherits this “marginalization” of the action costs from h_{CS} , and the semantics of this cost partition corresponds to a stronger than $|^+$ relaxation $|^{+s}$. In $|^{+s}$, we have $a \in A|^{+s}$ if and only if $a = (\text{pre}(a'), \{p\}, \emptyset)$ for some $a' \in A|^{+}$, $p \in \text{add}(a')$, and $C(a) = C(a')/|\text{add}(a)|$. In other words, the actions of $\Pi|^{+s}$ are obtained from the actions of Π by (i) removing their delete lists, (ii) splitting each action into a set of *single-effect* actions, and (iii) dividing the cost of each original action between the actions resulted from its splitting. It is not hard to verify that $h^{+s} \leq h^+$, yet computing h^{+s} is still NP-hard (follows from Theorem 4.2 and Corollary 4.3 in (Bylander 1994)). Proposition 4 connects between the $|^{+s}$ relaxation and computing h_{pmax} , showing that the latter basically aims at approximating the optimal plan cost for that relaxation.

Proposition 4 *Computing h_{pmax} over $\Pi|^{+}$ is equivalent to computing it over $\Pi|^{+s}$.*

Given that $\Pi|^{+s}$ is always at least as relaxed as $\Pi|^{+}$, at first view, the utility of targeting h^{+s} instead of h^+ in the process of approximation is unclear. However, this “step down” in h_{pmax} allows us for a more delicate treatment of interactions between the problem’s actions. For example, consider two Blocksworld problems involving n blocks b_1, \dots, b_n . In the first problem, the blocks are all initially unstacked, and the goal is to build an ordered tower with b_1 on top and b_n at the bottom. Assuming all actions have unit cost, for this initial state we have $h^+ = h_{\text{pmax}} = h_{\text{add}} = n - 1$, and $h_{\text{max}} = 1$. In the second problem, the blocks are initially stacked in an ordered tower with b_1 on top and b_n at the bottom, and the goal is to obtain a reversely ordered tower with b_n on top and b_1 at the bottom. For these state and goal, we have $h^+ = h_{\text{pmax}} = h_{\text{max}} = n$, and $h_{\text{add}} = 2 \sum_{i=1}^{n-1} i$. Note that the relative informativeness of h_{max} and h_{add} varies between these two problems, while h_{pmax} sticks to the correct value of h^+ . In fact, looking at the definition of h_{pmax} in Eqs. 8-10, one may wonder whether h_{pmax} is actually admissible. This, however, is not so, and the following example illustrates the pitfall. Let relaxed actions a_1, \dots, a_6 be defined as follows. For $1 \leq i \leq 3$, $\text{pre}(a_i) = \{p_i\}$, $\text{add}(a_i) = \{p_{i+3}\}$, and

$$\begin{aligned} \text{pre}(a_4) &= \{p_4, p_5\}, \text{add}(a_4) = \{p_7\} \\ \text{pre}(a_5) &= \{p_4, p_5\}, \text{add}(a_5) = \{p_8\} \\ \text{pre}(a_6) &= \{p_6\}, \text{add}(a_6) = \{p_7\} \end{aligned}$$

Given these action set, for $I = \{p_1, p_2, p_3\}$, and $G = \{p_7, p_8\}$, we have $h_{\text{pmax}}(I) = 5 > h^+(I) = 4$.

The question of whether h_{pmax} can be slightly modified to guarantee admissibility remains open. Likewise, before we move to discussing the empirical evaluation, it is worth noting that the idea of vector-based cost propagation can be taken beyond the specific setting of h_{pmax} . For instance, using cost vectors in $\mathcal{D} = \mathbb{R}^A$ (instead of in \mathbb{R}^P) allows us to provide in Eq. 11 a cost-sharing formulation of h_{FF} , required for Proposition 2.

$$\begin{aligned} \vec{w}_{a(t)}[a'] &= \max_{(p(t), a(t)) \in E_{a(t)}} \{ \vec{w}_{p(t)}[a'] \} \\ w_{a(t)} &= C(a(t)) + \sum_{a' \in A} \vec{w}_{a(t)}[a'] \\ \widehat{a(t)} &= \underset{(a(t), p(t+1)) \in E_{p(t+1)}}{\text{argmin}} \{ w_{a(t)} \} \\ \vec{w}_{p(t+1)} &= \begin{cases} \vec{0}, & t + 1 = 0 \\ \vec{w}_{\widehat{a(t)}}, & t + 1 > 0 \end{cases} \end{aligned} \quad (11)$$

Needless to say that computing h_{FF} this way is much more costly than doing that using the simple back-chaining procedure suggested in (Hoffmann & Nebel 2001). Moreover, while “noops-first” strategy from (Hoffmann & Nebel 2001) can be incorporated in Eq. 11, “action nodes reuseage” strategy used in computing the FF’s heuristic cannot be formulated within the cost-sharing framework. On the other hand, however, cost propagation might provide a better guidance on what no-noop action should better be selected to support achieving this or another sub-goal. Further insights into this tradeoff are required, and for now, h_{FF} formulation as in Eq. 11 is mostly of theoretical interest.

Evaluation

We have implemented h_{pmax} and (the original FF’s heuristic) h_{FF} within the HSP2 planner (Bonet & Geffner 2001) that have already supported the h_{add} and h_{max} heuristics. For the evaluation, we have used Strips problems from the IPC-2000 domains Blocksworld, Freecell, and Logistics, IPC-2002 domains Driverlog, Zenotravel, Depots, and Satellite, IPC-2004 domain Pipesworld, and IPC-2006 domains Openstacks, Pathways, Rovers, and TTP, and, finally, the n -Puzzle problems from the HSP2 distribution. As the quality of the generated solutions was of our interest, all the heuristics have been evaluated under the A^* search algorithm.

In general, the evaluation was aiming at answering the following questions.

- (1) How cost of the plans discovered by A^* with h_{pmax} compares to the optimal plans, as well as to the cost of the plans discovered with h_{add} and h_{FF} ?
- (2) Considering the runtime efficiency in terms of both the number of expanded nodes, and the computing time required per search node, how A^* with h_{pmax} compares to A^* with h_{max} , and to A^* with h_{add} and h_{FF} ?
- (3) To what extent the plan-quality effectiveness of h_{pmax} is attributed to the cost partitioning of $|^{+s}$, and to what extent to propagating vectors of costs?

Domain(#)	h_{add}	h_{FF}	h_{pmax}	h_{pmax}^\dagger
blocksworld (15)	7	15	15	15
logistics (10)	4	10	10	10
tpp (6)	2	5	6	6
pathways (4)	4	3	4	4
driverlog (9)	3	8	9	8
pipesworld (8)	5	5	8	6
zenotravel (8)	5	7	8	7
freecell (6)	4	5	6	5
depots (3)	2	2	3	2
rovers (4)	3	3	4	3
puzzle (4)	0	2	4	3
openstacks (3)	0	1	3	1
satellite (4)	3	3	3	3

Table 1: Quality-wise performance of A^* guided by different heuristics. A table entry (D, h) holds the number of optimal solutions discovered by A^* with h among all problems from D .

Considering question (1), the h_{add} , h_{FF} , and h_{pmax} columns in Table 1 summarize the effectiveness of these heuristics on guiding A^* toward optimal solutions. In general, h_{add} typically does not guide the search towards an optimal solution, and only in Pathways all the solutions discovered with h_{add} were optimal. h_{FF} is already much more effective, yet only for two domains it provided optimal solutions across all the problem instances that have been solved by A^* with both h_{FF} and h_{pmax} within the time limit of 1 hour. In contrast, the cost-sharing heuristic h_{pmax} appears to be almost as quality-wise effective as the admissible h_{max} —the only problem on which we found h_{pmax} guiding A^* to a non-optimal solution was instance #9 in the Satellite domain.

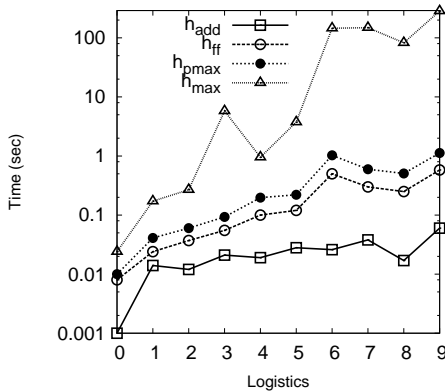


Figure 4: Runtime of A^* (in seconds) with h_{pmax} , h_{max} , and h_{add} , and h_{FF} heuristics on Logistics.

Considering question (2), first, Table 2 summarizes the relative efficiency of A^* in terms of the *amount of expanded*

Domain	h_{FF}	h_{pmax}	h_{max}	h_{pmax}^\dagger
blocksworld	6.1	46.8	266.3	1.8
logistics	11.2	11.2	>1945.2	11.2
tpp	0.7	7.9	53.3	0.7
pathways	2.17	9.8	3886.0	10.2
driverlog	19.8	26.6	>3042.0	21.6
pipesworld	1.6	188.1	510.0	1.4
zenotravel	0.8	12.9	>626.1	0.8
freecell	0.73	118.2	>1215	0.74
depots	12.3	89.8	>3462.0	2.2
rovers	0.9	8.7	62.4	0.9
puzzle	7.8	57.4	301.4	4.8
openstacks	1.7	53.5	76.0	6.3
satellite	20.3	11.2	>4277.4	11.2

Table 2: Relative efficiency in terms of number of expanded nodes, with h_{add} as a baseline. If $EN(p, h)$ is the number of nodes expanded by A^* guided by h on problem p , then, for each domain D , and each heuristic h' , the table entry $(D, h') = \text{avg}_{p \in D} \{EN(p, h')/EN(p, h_{\text{add}})\}$.

nodes. Let $EN(p, h)$ be the number of nodes expanded by A^* with heuristic h on problem p . Taking A^* with h_{add} as a reference point, for each domain D , and each heuristic $h \in \{h_{\text{FF}}, h_{\text{pmax}}, h_{\text{max}}\}$, the table provides the, averaged over D 's problem instances, ratio between $EN(p, h)$ and $EN(p, h_{\text{add}})$. (Here as well, the average is over problem instances on which A^* with both h_{FF} and h_{pmax} have finished within the time limit of one hour.) It is apparent from Table 2 that

- in all the domains, using h_{pmax} resulted in expanding less search nodes than using h_{max} , with the difference between the two reaching more than three orders of magnitude.
- using h_{pmax} resulted in expanding more search nodes than h_{FF} in all domains except for Satellite, with the difference between the two being mostly of up to one order of magnitude. In addition, we note that in 5 domains, namely Blocksworld, Logistics, Driverlog, Depots, and Satellite, the relation between h_{pmax} and h_{FF} in terms of the number of expanded nodes varied between the problem instances.

Now, the cost vector propagation clearly makes computing h_{pmax} per search node more costly than computing h_{FF} , and the difference between the two in this respect depends on the structure of the domain in hand. Figure 4, however, shows that this difference is not as substantial as one would probably conjecture. For instance, we noticed that in the Logistics domain h_{pmax} and h_{FF} provide exactly the same numbers of expanded and generated nodes across all the problem instances⁴. Thus, the time gap between A^* with h_{pmax} and h_{FF} in Logistics (depicted in Figure 4) is exclusively due

⁴In that sense, the Logistics domain was exceptional.

to the time difference in computing the heuristic values for each search node. Specifically, if $T(p, h)$ is the overall time taken by A^* with h on problem p , then we had

$$\text{avg}_{p \in \text{Logistics}} \{T(p, h_{\text{pmax}})/T(p, h_{\text{FF}})\} = 1.8$$

Finally, considering question (3), we have implemented a variant of h_{pmax} , referred in Tables 1-2 as $h_{\text{pmax}}^{\dagger}$, that eliminates the cost partitioning from Eqs. 9-10, all else being equal. Considering the number of “optimally solved domains” in Table 1, one can notice that A^* with $h_{\text{pmax}}^{\dagger}$ was still more effective than A^* with h_{FF} , and this using very similar numbers of expanded nodes (see Table 2). Thus, propagating compound cost quantities such as cost vectors does impact the plan-quality effectiveness of the search. However, Tables 1 also shows that A^* with $h_{\text{pmax}}^{\dagger}$ was *substantially* less effective than with h_{pmax} , bringing us to optimal solutions across 5 instead of 12 domains. We believe this provides a clear evidence for the marginal impact of the cost partitioning (that is, of using the $|^{+s}$ abstraction) on the informativeness of the h_{pmax} heuristic with respect to the optimal problem solving.

Summary and Future Work

We introduced a framework for approximating the optimal cost solutions for classical planning with no delete effects. This framework, called here cost-sharing, unified some previously suggested approximations, and suggested “importing” some admissible approximations developed for problems of probabilistic reasoning. While the latter import appeared not to be fruitful, its analysis suggested a way to develop novel approximations that, in particular, appeared to be more informative (in terms of guidance towards solutions of higher quality) than the seminal h_{add} and h_{FF} heuristics.

The cost propagation underlying the cost-sharing framework suggests numerous directions for further research.

- In this work, we focused on the “clean room” relaxation $|^{+}$ that completely ignores the delete effects of the actions. At the next step, it is only natural to look for a semantically-clean and effective way of extending the cost-sharing framework to (selectively) account for negative interactions between the actions. If developed, the effectiveness of such an extension could be compared to this of some state-of-the-art heuristics that also account for such type of information (e.g, the h^m heuristics (Haslum & Geffner 2000), the planning graph heuristics (Nguyen, Kambhampati, & Nigenda 2002), etc.)
- We believe that vector-based cost propagation has a potential in developing heuristics for problems with richer metrics of plan quality. In particular, currently we are looking into extending h_{pmax} to account for penalties and rewards for achieving certain facts (Bonet & Geffner 2006).
- Finally, the search for a more informative than h_{max} , admissible estimate for h^{+} remains challenging, and we hope that our work had shed some new light on this interesting problem.

Acknowledgments

The authors were partially supported by the C. Wellner Research Fund.

References

- Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *AIJ* 129(1-2):5-33.
- Bonet, B., and Geffner, H. 2006. Heuristics for planning with penalties and rewards using compiled knowledge. In *KR-06*.
- Bylander, T. 1994. The computational complexity of propositional STRIPS planning. *AIJ* 69(1-2):165-204.
- Charniak, E., and Husain, S. 1991. A new admissible heuristic for minimal-cost proofs. In *AAAI*, 446-451.
- Edelkamp, S. 2001. Planning with pattern databases. In *ECP*, 13-34.
- Fikes, R. E., and Nilsson, N. 1971. STRIPS: A new approach to the application of theorem proving to problem solving. *AIJ* 2:189-208.
- Haslum, P., and Geffner, H. 2000. Admissible heuristics for optimal planning. In *ICAPS*, 140-149.
- Haslum, P.; Bonet, B.; and Geffner, H. 2005. New admissible heuristics for domain-independent planning. In *AAAI*, 1163-1168.
- Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *JAIR* 14:253-302.
- Korf, R. E. 1985. Depth-first iterative-deepening: An optimal admissible tree search. *AIJ* 27(1):97-109.
- McDermott, D. V. 1999. Using regression-match graphs to control search in planning. *AIJ* 109(1-2):111-159.
- Nguyen, X.-L.; Kambhampati, S.; and Nigenda, R. S. 2002. Planning graph as the basis for deriving heuristics for plan synthesis by state space and CSP search. *AIJ* 135(1-2):73-123.
- Pearl, J. 1984. *Heuristics - Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley.
- Refanidis, I., and Vlahavas, I. P. 2001. The GRT planning system: Backward heuristic construction in forward state-space planning. *JAIR* 15:115-161.
- Rintanen, J. 2006. Unified definition of heuristics for classical planning. In *ECAI*, 600-604.
- Shimony, S.; Domshlak, C.; and Santos, E. J. 1997. Cost-sharing in Bayesian Knowledge Bases. In *UAI*, 421-428.