Heuristics for Planning with Action Costs

Emil Keyder[1] and Hector Geffner[2]

[1] Universitat Pompeu Fabra
Passeig de Circumvalació 8
08003 Barcelona Spain
keyder@upf.edu
[2] ICREA & Universitat Pompeu Fabra
Passeig de Circumvalació 8
08003 Barcelona Spain
hector.geffner@upf.edu

Abstract. We introduce a non-admissible heuristic for planning with action costs, called the *set-additive heuristic*, that combines the benefits of the *additive heuristic* used in the HSP planner and the *relaxed plan heuristic* used in FF. The set-additive heuristic h^s_a is defined mathematically and handles non-uniform action costs like the additive heuristic h_a, and yet like FF's heuristic h_{FF}, it encodes the cost of a specific *relaxed plan* and is therefore compatible with FF's helpful action pruning and its effective enforced hill climbing search. The definition of the set-additive heuristic is obtained from the definition of the additive heuristic, but rather than propagating the value of the best supports for a precondition or goal, it propagates the supports themselves, which are then combined by set-union rather than by addition. We report then empirical results on a planner that we call FF(h^s_a) that is like FF except that the relaxed plan is extracted from the set-additive heuristic. The results show that FF(h^s_a) adds only a slight time overhead over FF but results in much better plans when action costs are not uniform.

1 Motivation

The additive heuristic used in HSP [1] and the relaxed plan heuristic used in FF [2] are two of the best known heuristics in classical planning. While both are based on the delete-relaxation, the latter produces more accurate estimates along with information in the form of 'helpful actions' that is exploited in the 'enforced hill climbing' search, where non-helpful actions are ignored. Better estimates, helpful action pruning, and enforced hill climbing search are actually the three reasons that make FF a more effective planner than HSP [2]. The additive heuristic used in HSP, however, has some advantages as well. In particular, it is defined mathematically rather than procedurally, resulting in a formulation that handles non-uniform actions costs.

In this work, we introduce a new non-admissible heuristic for planning that we call the *set-additive heuristic*, that combines the benefits of the *additive* and *relaxed plan* heuristics. The set-additive heuristic h^s_a is defined mathematically

D. Borrajo, L. Castillo, and J.M. Corchado (Eds.): CAEPIA 2007, LNAI 4788, pp. 140–149, 2007.

and handles non-uniform action costs like the additive heuristic h_a, and yet like FF's heuristic h_{FF}, it encodes the cost of a specific *relaxed plan* and thus is compatible with FF's helpful action pruning and its effective enforced hill climbing search. The motivation is similar to the works in [3,4] which also aim to make the FF planner sensitive to cost information, yet rather than modifying the planning graph construction or extraction phases to take action costs into account, we modify the cost-sensitive additive heuristic to yield relaxed plans.

The paper is organized as follows. We first review the cost model and the definitions of the additive heuristic and the planning graph heuristic used by FF. We then introduce the new set-additive heuristic and present empirical results.

2 Planning Model and Heuristics

We consider planning problems $P = \langle F, I, O, G\rangle$ expressed in Strips, where F is the set of relevant atoms or fluents, $I \subseteq F$ and $G \subseteq F$ are the initial and goal situations, and O is a set of (grounded) actions a with preconditions, add, and delete lists $Pre(a)$, $Add(a)$, and $Del(a)$ respectively, all of which are subsets of F.

For each action $a \in O$, there is also a *non-negative cost* $cost(a)$. In classical planning this cost is assumed to be positive and uniform for all actions, normally equal to 1. In such a case, the cost of a plan is given by the number of actions in the plan. More generally, we take the cost $cost(\pi)$ of a plan $\pi = a_0, \ldots, a_n$ to be

$$cost(\pi) = \sum_{i=0,n} cost(a_i)$$

The search for plans is guided by heuristics that provide an estimate of the cost-to-go that are extracted automatically from the problem encoding P. Two of the most common heuristics are the *additive heuristic* used in the HSP planner [1] and the *relaxed plan heuristic* used in FF. Both are based on the delete-relaxation P^+ of the problem, and they both attempt to approximate the optimal delete-relaxation heuristic h^+ which is well-informed but intractable. Heuristics that are not based on the delete-relaxation and are admissible are used in Graphplan [5] and HSPr* [6]. These heuristics, however, are not as informed as their non-admissible counterparts.

We review some of these heuristics below. In order to simplify the definition of some of the heuristics, we introduce in some cases a new dummy *End* action with *zero cost*, whose preconditions $G_1, \ldots, G_n$ are the goals of the problem, and whose effect is a dummy atom G. In such cases, we will obtain the heuristic estimate $h(s)$ of the cost from state s to the goal, from the estimate $h(G; s)$ of achieving the 'dummy' atom G from s.

3 The Additive Heuristic

Since the computation of the optimal delete-free heuristic h^+ is intractable, HSP introduces a polynomial approximation where all subgoals are assumed to

be *independent* in the sense that they can be achieved with no 'side effects'. This assumption is false normally (as is that of the delete-relaxation) but results in a simple heuristic function $h_a(s) = h_a(G; s)$ that can be computed efficiently in every state s visited in the search:

$$h_a(p; s) = \begin{cases} 0 & \text{if } p \in s \\ \min_{a \in O(p)}[h_a(a; s)] & \text{otherwise} \end{cases}$$

where $h_a(p, s)$ stands for an estimate of the cost of achieving the atom p from s, $O(p)$ is the set of actions in the problem that add p, and

$$h_a(a; s) = cost(a) + \sum_{q \in Pre(a)} h_a(q; s)$$

stands for the cost of applying the action a after achieving its preconditions.

The additive heuristic, as its name implies, makes the assumption that the cost of achieving a set of atoms is equal to the *sum* of the costs of achieving each of the atoms separately. When this assumption is true, either because action preconditions and subgoals can be achieved with *no side effects*, or because the goal and action preconditions contain *one atom at most*, h_a is equal to h^+, and hence the additive heuristic is optimal in the delete relaxation. Most often this is not the case, yet as shown early in [7] and later in the HSP planner [1], the additive heuristic h_a can often guide the search for plans fairly well. Versions of the additive heuristic appear also in [8,3,9], where the cost of joint conditions in action preconditions or goals is set to the sum of the costs of each condition in isolation. The additive heuristic h_a for classical planning is obtained simply by setting the action costs $cost(a)$ all to 1 (except for the 'dummy' End action).

4 The Relaxed Planning Graph Heuristic

The planner FF improves HSP along two dimensions: the heuristic and the basic search algorithm. Unlike h_a, the **heuristic** h_{FF} used in FF makes no independence assumption for approximating h^+, instead computing one plan for P^+ which is not guaranteed to be optimal. This is done by a Graphplan-like procedure [5], which due to the absence of deletes, constructs a planning graph with no mutexes, from which a plan $\pi_{\mathrm{FF}}(s)$ is extracted backtrack-free [2]. The heuristic $h_{\mathrm{FF}}(s)$ is then set to $|\pi_{\mathrm{FF}}(s)|$. The basic **search procedure** in FF is not *WA** as in HSP but (enforced) *hill-climbing* (EHC), in which the search moves from the current state s to a neighboring state s' with smaller heuristic value by performing a *breadth first search.* This breadth first search is carried out with a *reduced branching factor* where actions a that are not found to be 'helpful' in a state s are ignored. The 'helpful actions' in a state s are the actions applicable in s that add a relevant subgoal p, as judged from the computation of the relaxed plan $\pi_{\mathrm{FF}}(s)$. The more accurate relaxed plan heuristic, along with the reduced branching factor in the breadth first search that follows from the exclusion of non-helpful actions, make the FF planner scale up better than HSP [2].

An advantage of HSP over FF, however, is the ability to naturally take into account non-uniform actions costs. While the additive heuristic h_a extends naturally to such cases, the relaxed plan extraction procedure and the layered planning graph construction on which it is based do not. Some recent attempts to modify the planning graph construction in order to take cost information into account can be found in [3,4]. Here we take a different approach that avoids planning graphs entirely, relying instead on a simple modification of the additive heuristic to compute *relaxed plans*.

The new *set-additive heuristic* modifies the formulation of the additive heuristic slightly, so that rather than expressing *numbers* $h_a(p;s)$ it expresses 'relaxed plans' $\pi_a(p;s)$, i.e., sets of actions that can be ordered into plans for p from the state s in the delete-relaxation P^+.

5 The Set-Additive Heuristic

The definition of the additive heuristic can be rewritten as

$$h_a(p;s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } p \in s \\ h_a(a_p;s) & \text{otherwise} \end{cases}$$

where

$$a_p = \text{argmin}_{a \in O(p)} h_a(a;s)$$

is the *best supporting action* for p in s, and $h(a;s)$ is

$$h_a(a;s) = cost(a) + \sum_{q \in Pre(a)} h_a(q;s)$$

In the additive heuristic, the value of the best supporter a_p of p in s, $h_a(a_p;s)$, is propagated into the heuristic value of p, $h_a(p;s)$. The *set-additive* heuristic can be understood in terms of a small change: rather than propagating *the value* $h_a(a_p;s)$ of the best supporter a_p of p, it propagates *the supporter* a_p itself. In addition, unlike values, such supports are not combined by *sums* but by *set-unions*, resulting in a function $\pi_a(p;s)$ that represents a *set of actions*, which in analogy to $h_a(p;s)$ is defined as:[1]

$$\pi_a(p;s) = \begin{cases} \{\} & \text{if } p \in s \\ \pi_a(a_p;s) & \text{otherwise} \end{cases}$$

where

$$a_p = \text{argmin}_{a \in O(p)} Cost(\pi_a(a;s))$$

$$\pi_a(a;s) = \{a\} \bigcup \{\cup_{q \in Prec(a)} \pi_a(q;s)\}$$

[1] The value of the set-additive heuristic $h^s_a(s)$, unlike the value of the normal additive heuristic, depends on the way ties are broken. We assume that among several supports a_p with the same costs $Cost(a_p;s)$, the one containing fewer actions, i.e., smallest $|\pi_a(a_p;s)|$, is preferred.

$$Cost(\pi_a(a;s)) = \sum_{a' \in \pi_a(a;s)} cost(a')$$

That is, the best supporter a_p for p is propagated into p, and supports for joint preconditions and goals are combined by set-union. The best-supporter is selected in turn as the action a_p for which the 'plan' made up of the supports of each of its preconditions along with the action itself has minimum cost. The *set-additive heuristic* $h^s_a(s)$ for the state s is then defined as

$$h^s_a(s) = Cost(\pi_a(G;s))$$

While $\pi_a(p;s)$ is a *set* and not a *sequence* of actions, its definition ensures that the actions it contains can be ordered into an action sequence that is a *plan* for p in the relaxed problem P^+ from state s. Indeed, one such parallel plan can be obtained by scheduling in a 'first layer' A_0, the actions a in $\pi_a(p;s)$ with empty supports; i.e., with $\pi_a(a;s) = \{\}$, then in a 'second layer', the actions a with supports in the first layer only, i.e., with $\pi_a(a;s) \subseteq A_0$, and so on. Within each layer, the actions can be serialized in any way as there are no deletes in P^+. As a result, and provided that there is a (relaxed) plan for each atom p in the delete-relaxation P^+,[2] we have that:

Proposition 1. $\pi_a(p;s)$ *represents a* relaxed plan *for* p *from* s.

This means that $\pi_a(G;s)$ for the dummy goal G can play the role of the *relaxed plan* in FF in place of the planning graph extraction procedure that is not sensitive to cost information. The rest of FF's machinery, such as helpful actions, enforced hill climbing, and so on, can be kept in place. We will call the resulting planner FF(h^s_a).

Notice that since $\pi_a(G;s)$ is a *set* of actions, there are *no action duplicates* in the corresponding relaxed plan. This property is true also of the relaxed plan computed by FF, following from the NO-OP first heuristic [2].[3]

We have implemented the set-additive heuristic h^s_a on top of the code that computes the normal additive heuristic h_a in HSP, which is a Bellman-Ford algorithm that solves shortest-path problems [10,11,12]. For the set-additive heuristic, the label of a 'node' p in the graph must represent both the set of actions $\pi_a(p;s)$ and its cost $Cost(\pi_a(p;s))$. The sets of actions are represented as sparse, ordered lists so that the union of two such sets is done in time linear in the sum of their sizes. In the additive heuristic, the analogous operation is a sum which is certainly cheaper, yet as the experiments below show the computational cost of these unions is not prohibitive.

[2] This condition is easily enforced by adding 'dummy' actions a'_p with very high cost that add p for each p. Thus, if $h(p;s)$ is $h(a'_p)$, it means that there is no plan for achieving p from s in the relaxation.

[3] No action duplicates are needed in plans for the delete-relaxation of Strips problems. For problems involving conditional effects, however, this is no longer true. For applying the set-additive heuristic in such cases, conditional effects must be compiled exactly or approximately into action preconditions.

6 Additive and Set-Additive Heuristics Compared

The normal additive heuristic can be understood as the *bag* or *multiset* additive heuristic, which is exactly like the set-additive heuristic above, but with the expressions $\pi_a(p; s)$ combined as *bags* or *multisets* rather than *sets* [13]. A *bag* or *multiset* is a collection *with repetitions*, where each element can have a *multiplicity* greater than 1. E.g., in the the multiset $A = \{a, a, a, b, b, c\}$, the element a has multiplicity 3, b has multiplicity 2, and c has multiplicity 1 (all other elements have multiplicity 0). If $B = \{a, c\}$ is another multiset, then the multi-set union of A and B, is $\{a, a, a, a, b, b, c, c, \}$. If $\pi_a(p; s)$ is a *multiset*, then it may include duplicate actions that lead to *overcounting* when the costs of each of the actions in the multiset are added up. From this perspective, the *set-additive* heuristic eliminates the overcounting that arises from the *multiset-additive* heuristic, which is equivalent to the normal additive heuristic, by replacing *multisets* by *sets*. The result is a heuristic that like h_{FF} does not 'overcount' [2] and that like h_a is sensitive to cost information.

7 The FF(h^s_a) Planner

The FF(h^s_a) planner analyzed below is FF but with the relaxed plan $\pi_{\mathrm{FF}}(s)$ computed from the relaxed planning graph replaced by the one computed with the *set-additive heuristic*: $\pi_a(G; s)$. The resulting heuristic $h^s_a(s)$ is thus cost-sensitive, and furthermore, remains optimal in problems in which the normal additive heuristic is optimal as well, such as when preconditions and goals involve one atom at most. Two other small changes have been made to take action costs into account in the enforced hill-climbing procedure (EHC).

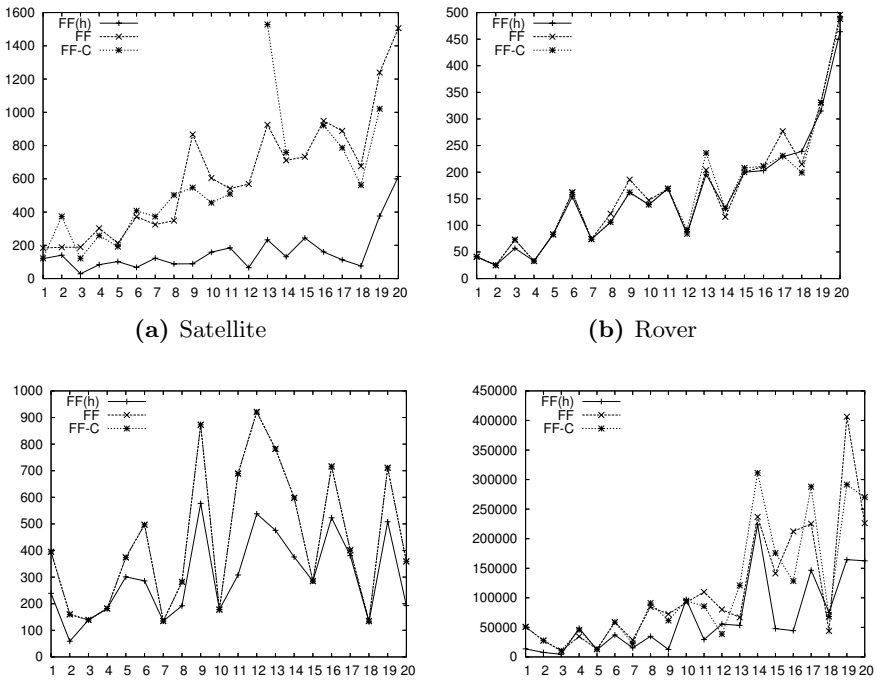
First, while a single step of EHC in FF ends as soon as a state s' is found by breadth-first search from s such that $h(s') < h(s)$, in FF(h^s_a) this is not true in the first level of the search. Instead, all states s' resulting from applying a helpful action a in s are evaluated, and among those for which $h(s') < h(s)$ holds, the action minimizing the expression $cost(a) + h(s')$ is selected.[4]

Second, while helpful actions in FF are defined as $H(s) = \{a \in A | add(a) \cap G_1 \neq \emptyset\}$, where G_1 denotes the set of atoms in the first layer of the planning graph arising from the extraction of the plan $\pi_{\mathrm{FF}}(s)$, in FF(h^s_a), G_1 is defined as the set of atoms p achievable in one step, i.e., $|\pi_a(p; s)| = 1$, such that p is a precondition of some action in the relaxed plan $\pi_a(G; s)$.

8 Experimental Results

We tested three heuristics in combination with two search algorithms. The heuristics are the additive heuristic h_a, the set-additive heuristic, and FF's

[4] Actually, when an action a maps s into a state s' in the first level such that $h(s') = h(s) - cost(a)$ and the size of the computed relaxed plan is decreased by 1, such an action is selected right away.



(a) Satellite **(b)** Rover

(c) Costgrid **(d)** Zeno

Fig. 1. Plan costs with EHC Search: FF, FF-C, and FF(h^s_a)

heuristic h_{FF}. The search algorithms are EHC and WA* with evaluation function $f(n) = g(n) + Wh(n)$ with $W = 5$. The only combination we did not try was h_a with EHC, as EHC requires the notion of a relaxed plan that the heuristic h_a does not provide.

The five combinations were implemented on top of Metric-FF, an extension of the FF planner that handles numeric fluents [14]. This is because the current accepted syntax for non-uniform action costs is expressed through numeric fluents and metric expressions that Metric-FF can handle. Numeric fluents, however, are only used to encode such cost information, and once the cost information is obtained from the input, numeric fluents are eliminated from the problem, leaving a boolean planning problem with cost information.

Experiments were performed with six domains with non-uniform action costs and five STRIPS domains. Four of these were versions of the domains Satellite, Rovers, Depots, and Zenotravel from the Third International Planning Competition (IPC3), modified as discussed above. The fifth domain, Driverlog, needed no modification as no numeric variables occur in preconditions of actions or goals. The sixth domain, Costgrid, is a simple grid domain in which movements between squares are randomly assigned costs between 0 and 100. It is possible to prove that in such a domain, the additive and set-additive heuristics are optimal as preconditions and goals involve a single atom. The five STRIPS domains used were the STRIPS versions of the five IPC3 domains.

All experiments were run on a grid consisting of 76 nodes, each a dual-processor Xeon "Woodcrest" dual core computer, with a clock speed of 2.33 GHz and 8 Gb of RAM. Execution time was limited to 1,800 seconds.

FF vs. FF(h^s_a): Quality. EHC with the set-additive heuristic often yields better plans than with FF's heuristic. This can be seen from the curves in Figure 1 that display plan costs over four domains. The differences are significant in Satellite, Zeno, and CostGrid, where we have found that the heuristic values computed by the two heuristics in the initial state are also the most different. On the other hand, the values produced by the two heuristics in Rovers, Depots, and Driverlog are closer, leading to plans with similar costs.

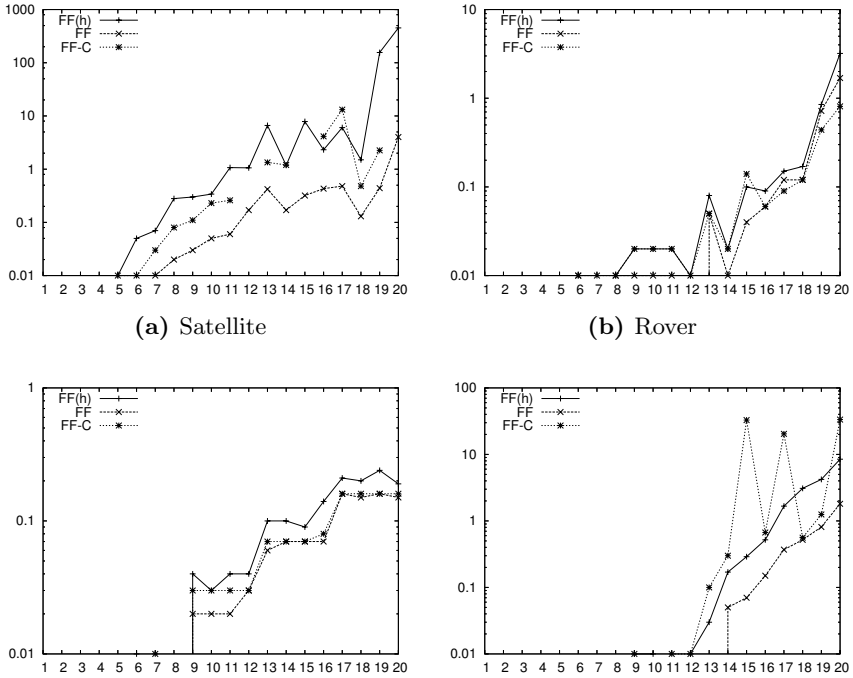
FF vs. FF(h^s_a): Time. FF(h^s_a) often takes longer than normal FF. The reasons are two: the overhead of propagating sets in the heuristic computation, and the fact that the plans that FF(h^s_a) finds are sometimes longer but with better overall cost. The times for the four domains above are shown in Figure 2. This overhead, however, does not affect coverage: FF and FF(h^s_a) in EHC mode solve all 20 instances of Satellite, Rovers, Zenotravel, and Costgrid, and both fail only in 2 of the 22 instances in Depots, and in 3 and 4 instances respectively of Driverlog.

FF with Costs vs. FF(h^s_a). Aside from the curves for FF and FF(h^s_a), Figures 1 and 2 show a third curve. This curve, labeled FF-C, corresponds to the combination of the modified EHC procedure used in FF(h^s_a) with a version of the FF heuristic that takes action costs into account. While $h_{\mathrm{FF}}(s)$ is $|\pi_{\mathrm{FF}}(s)|$, where $\pi_{\mathrm{FF}}(s)$ is the relaxed plan computed by FF from s, the heuristic $h^c_{\mathrm{FF}}(s)$ used in FF-C is the result of adding up the cost of the actions in $\pi_{\mathrm{FF}}(s)$. As it can be seen from the curves, FF-C improves FF in terms of plan quality in a few cases but not as often as FF(h^s_a) and not as much. This is because the relaxed plan extraction remains cost-insensitive. At the same time, FF-C is slower than FF, which by ignoring action costs completely, searches for the goal more greedily.

Heuristics in WA* Search. When the heuristics h_a, h^s_a, and h_{FF} are used in the context of the WA* search, the first two heuristics do better than the latter one. The coverage of the three heuristics is shown in Table 1, where the additive heuristic h_a does slightly better than the set-additive heuristic h^s_a (because it is cheaper to compute), and both do better than h_{FF}. On the other hand, the set-additive heuristic with EHC solves many more problems than the additive heuristic with WA*.

Table 1. Coverage of the three heuristics combined with a WA* search. There are 20 problems in each domain except for Depots with 22.

h	Satellite	Rovers	Zenotravel	Depots	Driverlog	Costgrid
h_a	0	4	14	13	11	20
h^s_a	0	4	11	13	9	20
h_{FF}	0	5	8	10	6	20



(a) Satellite **(b)** Rover

(c) Costgrid **(d)** Zeno

Fig. 2. Times in EHC Search: FF, FF-C, and FF(h^s_a)

Uniform vs. Non-Uniform Costs. The heuristic values computed by h^s_a and h_{FF} when costs are uniform are not necessarily equal, yet we have found them to be very much alike over these domains, leading to plans with roughly the same costs. We omit the corresponding graphs due to lack of space. This suggests that when costs are uniform, the overhead in the computation of the set-additive heuristic does not pay off. For non-uniform costs, on the other hand, h^s_a used with EHC search appears to yield the best tradeoff.

9 Summary

We have introduced a new non-admissible heuristic for planning, the *set-additive heuristic*, that combines the benefits of the *additive* and *relaxed plan* heuristics. The motivation is similar to the work in [3,4] which also aims to make the FF planner sensitive to cost information, but rather than modifying the plan graph construction or extraction phase to take action costs into account, we have modified the cost-sensitive additive heuristic to yield relaxed plans. The resulting formulation sheds light also on the normal additive heuristic, which can now be as the *multiset-additive* heuristic, and suggests further refinements that can result from the propagation of symbolic labels (supports) rather than numbers in the basic formulation.

Acknowledgements

We thank the anonymous reviewers for useful comments. H. Geffner is partially supported by Grant TIN2006-15387-C03-03 from MEC, Spain.

References

1. Bonet, B., Geffner, H.: Planning as heuristic search. Artificial Intelligence 129(1–2), 5–33 (2001)
2. Hoffmann, J., Nebel, B.: The FF planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research 14, 253–302 (2001)
3. Sapena, O., Onaindia, E.: Handling numeric criteria in relaxed planning graphs. In: Lemaître, C., Reyes, C.A., González, J.A. (eds.) IBERAMIA 2004. LNCS (LNAI), vol. 3315, pp. 114–123. Springer, Heidelberg (2004)
4. Fuentetaja, R., Borrajo, D., Linares, C.: Improving relaxed planning graph heuristics for metric optimization. In: Proc. 2006 AAAI Workshop on Heuristic Search, Memory Based Heuristics and its Applications, pp. 79–86 (2006)
5. Blum, A., Furst, M.: Fast planning through planning graph analysis. In: Proceedings of IJCAI 1995, pp. 1636–1642. Morgan Kaufmann, San Francisco (1995)
6. Haslum, P., Geffner, H.: Admissible heuristics for optimal planning. In: Proc. of the Fifth International Conference on AI Planning Systems (AIPS-2000), pp. 70–82 (2000)
7. Bonet, B., Loerincs, G., Geffner, H.: A robust and fast action selection mechanism for planning. In: Proceedings of AAAI 1997, pp. 714–719. MIT Press, Cambridge (1997)
8. Do, M.B., Kambhampati, S.: Sapa: A domain-independent heuristic metric temporal planner. In: Proc. ECP 2001, pp. 82–91 (2001)
9. Smith, D.E.: Choosing objectives in over-subscription planning. In: Proc. ICAPS 2004, pp. 393–401 (2004)
10. Bertsekas, D.: Linear Network Optimization: Algorithms and Codes. MIT Press, Cambridge (1991)
11. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. MIT Press, Cambridge (1989)
12. Liu, Y., Koenig, S., Furcy, D.: Speeding up the calculation of heuristics for heuristic search-based planning. In: Proc AAAI 2002, pp. 484–491 (2002)
13. Blizard, W.D.: Multiset theory. Notre Dame J. Formal Logic 30(1), 36–66 (1988)
14. Hoffmann, J.: The metric-ff planning system: Translating "ignoring delete lists" to numeric state variables. J. Artif. Intell. Res (JAIR) 20, 291–341 (2003)