Set-Additive and TSP Heuristics for Planning with Action Costs and Soft Goals

Emil Keyder
Universitat Pompeu Fabra
Passeig de Circumvalació 8
08003 Barcelona Spain
emil.keyder@upf.edu

Héctor Geffner
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 is defined mathematically and handles non-uniform action costs like the additive heuristic but like FF's heuristic, it encodes the cost of a specific *relaxed plan* and hence is compatible with FF's helpful action pruning and its enforced hill climbing search. The set-additive heuristic changes the definition of the additive heuristic slightly by associating with each atom a relaxed plan rather than its numeric cost. The new formulation is used then to introduce a further variation that takes certain delete information into account by forcing the values of certain multivalued variables in the relaxed plan to be spanned by a path rather than by a tree. Finally, we show how soft goals can be compiled away and report empirical results using a modification of the FF planner that incorporates these ideas, leading to a planner that is as robust as FF but capable of producing better plans in a broader set of contexts.

Motivation

The additive heuristic used in HSP (Bonet & Geffner 2001) and the relaxed plan heuristic used in FF (Hoffmann & Nebel 2001) 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' (EHC) search. The additive heuristic, however, has some advantages as well; it is defined mathematically rather than procedurally, and takes non-uniform action costs naturally into account.

In this work, we aim to combine the benefits of the *additive* and *relaxed plan* heuristics in a new non-admissible heuristic for planning that we call the *set-additive heuristic*. The set-additive heuristic h^s_a is defined mathematically and accounts naturally for non-uniform action costs, but 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 for this extension is similar to the works in (Sapena & Onaindia 2004; Fuentetaja, Borrajo, & Linares) that also aim to make the FF planner sensitive to cost information, but *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.*

In addition, we use the formulation of the new heuristic to introduce a variation that overcomes some of the limitations of delete-based relaxations by forcing the values of certain *multivalued variables* in the relaxed plan to be spanned by a *path* rather than by a *tree*. For example, in a problem where a number of rocks have to be collected from various locations $l_1, \ldots, l_n$ starting from a location l_0, an action sequence where the agent repeatedly moves from l_0 to each location l_i, $i = 1, \ldots, n$ is not possible in the original problem but is possible in the delete-relaxation. The result is that the cost of such problems is approximated in terms of the *min cost tree* rooted at l_0 that spans all the locations l_i. The new heuristic, on the other hand, approximates the cost of such problems in terms of the cost of the *best path* that visits all such locations, a problem that corresponds to a Traveling Salesman Problem. Unlike other recent proposals that rely on quick but suboptimal TSP algorithms for planning however (Long & Fox 2000; Smith 2004), our approach integrates such algorithms in the computation of the domain-independent heuristic that is defined in a declarative way.

Finally, we show how these cost-sensitive heuristics can be used without change in the context of *soft goals*: atoms that produce a positive reward if achieved by the end of the plan (Smith 2004; Sanchez & Kambhampati 2005). In order to achieve this, we show how *soft goals can be compiled away* leaving an equivalent problem with an extended set of fluents and actions but no soft goals. Some preliminary experiments are reported as well.

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 precondition, 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 a *non-negative cost* $cost(a)$ and we are interested in plans π that minimize the

cost

$$cost(\pi) = \sum_{i=1,n} cost(a_i) \ . \quad (1)$$

This problem is a generalization of the problem in classical planning where $c(a)$ is uniform and equal to 1, and $c(\pi)$ measures the number of actions in the plan. We are not interested however in optimal planning according to this metric, but in finding good plans using heuristics that are sensitive to the cost information.

Two of the most common heuristics in planning are the *additive heuristic* used in HSP (Bonet & Geffner 2001) and the *relaxed plan heuristic* used in FF. Both are based on the delete-relaxation P^+ of the problem, and both attempt to approximate the optimal delete-relaxation heuristic h^+ which is well-informed but intractable. We review them 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 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.

The Additive Heuristic

Since the computation of the optimal delete-free heuristic h^+ is intractable, HSP introduces a polynomial approximation in which all subgoals are assumed to be *independent.* This assumption is normally false (as is the delete-relaxation) but results in a simple heuristic function

$$h_a(s) \stackrel{\text{def}}{=} h(G; s) \quad (2)$$

that can be computed quite efficiently in every state s visited in the search defined as:

$$h(p; s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{if } p \in s \\ \min_{a \in O(p)}[h(a; s)] & \text{otherwise} \end{cases} \quad (3)$$

where $h(p; s)$ 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

$$h(a; s) \stackrel{\text{def}}{=} cost(a) + \sum_{q \in Pre(a)} h(q; s) \quad (4)$$

stands for the cost of achieving the preconditions of an action a and applying it.

The additive heuristic is optimal for P^+ when the goal and preconditions include one atom at most, where it coincides with the h_{max} heuristic (Bonet & Geffner 2001). Versions of the additive heuristic appear in (Do & Kambhampati 2001; Sapena & Onaindia 2004; Smith 2004), 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 Relaxed Planning Graph Heuristic

The planner FF improves HSP along two dimensions, the heuristic and the search algorithm. Unlike h_a, the heuristic h_{FF} used in FF makes no independence assumption for approximating h^+, computing instead one plan for P^+ which is not guaranteed to be optimal. This is done by a Graphplan-like procedure (Blum & Furst 1995), which due to the absence of deletes, constructs a planning graph with no mutexes, from which a plan $\pi_{\text{FF}}(s)$ is extracted backtrack-free (Hoffmann & Nebel 2001). The heuristic $h_{\text{FF}}(s)$ is then set to $|\pi_{\text{FF}}(s)|$. Likewise, the search procedure in FF is not *best-first* 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*, ignoring actions a that are not found to be 'helpful' in state s. The 'helpful actions' in a state s are the actions applicable in s that add a relevant subgoal p, as determined during the computation of the relaxed plan $\pi_{\text{FF}}(s)$.

The more accurate planning graph heuristic, along with the reduced branching factor in the breadth first search, makes the FF planner scale up better than HSP (Hoffmann & Nebel 2001). On the other hand, the heuristic h_{FF} is not cost-sensitive. A possibility for making FF sensitive to cost information is to move to an alternative heuristic h^c_{FF} defined as the sum of the action costs $cost(a)$ over the actions in the relaxed plan π_{FF}. Yet, this move does not solve the problem that the computation of the relaxed plan itself ignores the cost information. We will say more about the heuristics h_{FF} and h^c_{FF} in the experimental section.

The Set-Additive Heuristic

The *set-additive heuristic* is a simple modification of the additive heuristic that produces relaxes plans taking the cost information into account. The definition of the additive heuristic can be rewritten as

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

where

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

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

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

In the additive heuristic, the *value* $h(a_p; s)$ of the best supporter a_p for p is propagated to obtain the heuristic value $h(p; s)$. In the *set-additive* heuristic, the *best supporter* itself is propagated, resulting in a function $\pi(p; s)$ that represents a *set of actions*, which can be defined similarly to $h(p; s)$ as [1]:

$$\pi(p; s) = \begin{cases} \{\} & \text{if } p \in s \\ \pi(a_p; s) & \text{otherwise} \end{cases} \quad (5)$$

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

where

$$a_p = \text{argmin}_{a \in O(p)} Cost(\pi(a;s)) \quad (6)$$

$$\pi(a;s) = \{a\} \bigcup \{\cup_{q \in Pre(a)} \pi(q;s)\} \quad (7)$$

$$Cost(\pi(a;s)) = \sum_{a' \in \pi(a;s)} cost(a') \quad (8)$$

That is, the best supporter a_p of p is propagated to compute $\pi(p;s)$ and supports for preconditions are combined by set-union rather than by sum. We will see that the set of actions $\pi(p;s)$ stands actually for a relaxed plan that achieves p from s, while $\pi(a;s)$ is a relaxed plan that achieves each of the preconditions of a and applies then a.

The *set-additive heuristic* $h^s_a(s)$ for a state s is then defined as

$$h^s_a(s) = Cost(\pi(G;s)) \quad (9)$$

Though $\pi(p;s)$ 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 the start state s. One such parallel plan can be obtained by scheduling in a 'first layer' A_0 the actions a in $\pi(p;s)$ with empty supports $\pi(a;s) = \{\}$, in a 'second layer' A_1 the actions a with supports in the first layer only, i.e., with $\pi(a;s) \subseteq A_0$, and so on.

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

This means that $\pi(G;s)$ 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, i.e. helpful actions, enforced hill climbing, and so on, can be kept in place, as long as the relaxed plan computed in each state s is the one encoded by $\pi(p;s)$.

Note that since $\pi(G;s)$ is actually a *set* of actions, there are *no action duplicates* in the corresponding relaxed plan. This property is also true of the relaxed plan computed by FF, following from the NO-OP first heuristic (Hoffmann & Nebel 2001). The normal additive heuristic can actually be understood as a version of the set-additive heuristic in which relaxed plans $\pi(p;s)$ are used to denote *bags* or *multisets* rather than *sets*, leading to a single action being counted many times.

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 for solving a shortest-path problem (Bertsekas 1991; Cormen, Leiserson, & Rivest 1989). For the set-additive heuristic, the label of a 'node' p in the graph must represent both the set of actions $\pi(p;s)$ and its cost $Cost(\pi(p;s))$. 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. The analogous operation for the additive heuristic is a sum. While this is certainly cheaper, as the experiments below show, the computational cost of the unions is not prohibitive.

Evaluating the h^s_a heuristic

We tested the heuristics h_{FF}, h^c_{FF}, and h^s_a in the context of an EHC search. The first two heuristics are based on the relaxed plan extracted by FF with the difference that the first counts the number of actions in the plan, while the latter adds up their costs. The last is the set-additive heuristic where the relaxed plan extraction itself is cost-sensitive. We refer to the resulting planners as FF, FF-C, and FF(h^s_a) respectively.

The EHC search with the *set-additive heuristic* uses $\pi(G;s)$ as the relaxed plan from s. In addition, due to the costs involved, two changes are done with respect to FF. First, while a single step of EHC in FF ends as soon as a state s' is found by a breadth-first search from s such that $h(s') < h(s)$, in FF(h^s_a) (and in FF-C), 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.[2] 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 $\pi_{FF}(s)$, in FF(h^s_a), G_1 is defined as the set of atoms p achievable in one step, i.e., $|\pi(p;s)| = 1$, such that p is a precondition of some action in the relaxed plan $\pi(G;s)$.

The three planners above were implemented on top of Metric-FF, an extension of the FF planner (Hoffmann 2003). This is because the current accepted syntax for non-uniform action costs is expressed through numeric fluents that Metric-FF can handle. Numeric fluents, however, are used only to encode cost information, and are then pruned away.

The experiments were performed over six domains: four are modified versions of the numeric domains Satellite, Rovers, Depots, and Zenotravel from the Third International Planning Competition (IPC3). In all cases, once the action costs $c(a)$ are extracted from the problem, all the numeric variables are eliminated, leaving us with a Strips planning problem with non-uniform costs. 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.

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.

We have found that the set-additive heuristic h^s_a yields better plans than both h_{FF} and h^c_{FF}, the differences being significant in Satellite, Zeno, and Costgrid. In terms of search time, search with h^s_a takes longer than with h_{FF}, because it is more expensive to compute and because sometimes the better plans are simply longer. Yet the difference in time in the experiments is constant factor that ranges between 4 and 10 with both searches scaling up roughly in the same way. Figures 1 and 3 show the quality of the plans for Satellite

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

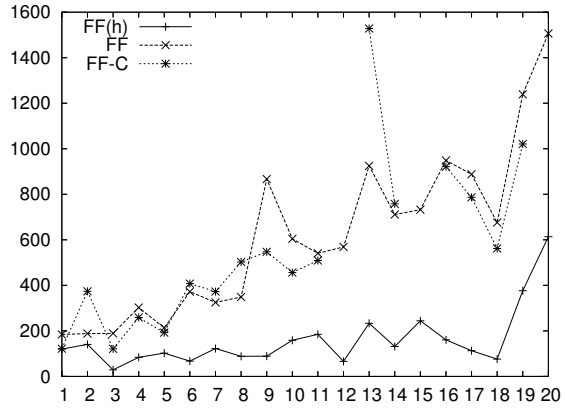

Figure 1: Plan costs for Satellite ($h = h^s_a$)

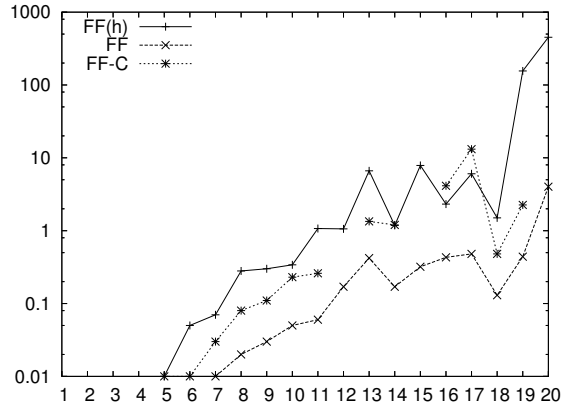

Figure 2: Plan times for Satellite ($h = h^s_a$)

and Zeno, while Figure 2 shows the search time for Satellite. The heuristic h^c_{FF} is not a clear improvement over the normal cost insensitive heuristic h_{FF}, and this is probably why this heuristic is not used in Metric-FF.

We have also done experiments in domains with *uniform costs*. In such domains the heuristic values returned by h_{FF} and h^s_a are very much alike, and since the latter is more expensive to compute, its use does not appear to be justified. Both heuristics also produce estimates that are lower than the normal additive heuristic that tends to overestimate as a result of the action duplicates.

Richer Labels and the TSP Heuristic

The additive heuristic $h_a(s)$ is defined by associating a number $h_a(p; s)$ to every atom p in the problem (Equations 2–4). The set-additive heuristic $h^s_a(s)$ on the other hand associates with every atom a relaxed plan $\pi(p; s)$ (Equation 5–9). The latter can be generalized by replacing the plans $\pi(p; s)$ with more generic **labels** $L(p; s)$ that can be numeric, symbolic, or a suitable combination, provided that there is a function $Cost(L(p; s))$ that maps labels $L(p; s)$ to numbers.

Here we consider labels $L(p; s)$ that result from treating

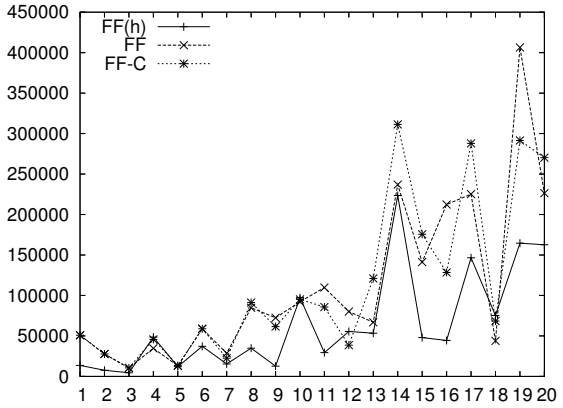

Figure 3: Plan costs for Zeno ($h = h^s_a$)

one designated multivalued variable X in the problem in a special way. A multivalued variable X is a set of atoms x_1, ..., x_n such that exactly one x_i holds in every reachable state. For example, in a task where there are n rocks r_1, ..., r_n to be picked up at locations l_1, ..., l_n, the set of atoms $at(l_0)$, $at(l_1)$, ..., $at(l_n)$, where $at(l_0)$ is the initial agent location, represent one such variable, encoding the possible locations of the agent. If the cost of going from location l_i to location l_k is $c(l_i, l_k)$, then the cost of picking up all the rocks is the cost of the best (min cost) *path* that visits all the locations, added to the costs of the pickups. This problem is a TSP that is intractable but whose cost can be approximated by quick and suboptimal TSP algorithms. The delete-relaxation on the other hand approximates the model of the problem by relating its cost to the cost the best *tree* rooted at t_0 that spans all the locations. The modification of the labels $\pi(a; s)$ in the set-additive heuristic allows us to move from the *approximate model* captured by the delete-relaxation to *approximate TSP algorithms* over a more accurate model.

For this, we assume that the actions that affect the selected multivalued variable X do not affect other variables in the problem, and maintain in the labels $\pi(p; s)$ two disjoint sets: a set of *actions* that do not affect X, and the set of *X-atoms* required by these actions. The heuristic $h_X(s)$ is then defined as

$$h_X(s) \stackrel{\text{def}}{=} Cost_X(\pi(G; s)) \tag{10}$$

where $Cost_X(S)$ is the sum of the action costs for the actions in S that do not affect X plus the estimated cost of the 'local plan' (Brafman & Domshlak 2006) that generates all the X-atoms in S, expressed as $TSP_X(S \cap X; s)$ and to be defined below:

$$Cost_X(S) = \sum_{a \in (S \cap \bar{X})} cost(a) + TSP_X(S \cap X; s) \tag{11}$$

The equations for the labels $\pi(p; s)$ follow the ones in the set-additive heuristic with the difference that *no commitment is made about the supports of X-atoms*; the required X-atoms are accumulated in the label instead, with their supports left to be determined by the TSP algorithm. As a result,

the equations for atoms $p \notin X$ and actions a that do not affect X are:[3]

$$\pi(p; s) = \begin{cases} \{\} & \text{if } p \in s \\ \pi(a_p; s) & \text{otherwise} \end{cases} \quad (12)$$

where

$$a_p = \text{argmin}_{a \in O(p)} Cost_X(\pi(a; s))$$
$$\pi(a; s) = \{a\} \cup \{\cup_{q: X^-(a)} \pi(q; s)\} \cup \{\cup_{x: X^+(a)} \{x\}\}$$

with $X^-(a) = \{Pre(a) \cap \bar{X}\}$ and $X^+(a) = \{Pre(a) \cap X\}$.

Finally, the estimate $TSP_X(V; s)$ of the cost of the best action sequence that generates all the atoms $x_i \in V$ starting from s (such atoms are mutex and cannot be achieved jointly) is obtained by a fast but suboptimal TSP algorithm over a directed graph with vertex set $V' = V \cup \{x_s, x_d\}$, where x_s is the X-atom that is true in s and x_d is a dummy vertex. The edges in this graph are the pairs (x_i, x_j) of 'vertices' x_i and x_j in V', $x_i \neq x_j$, and their costs $c_{i,j}$ is

$$c_{i,j} = \begin{cases} D_s(x_i, x_j) & \text{if } x_i \neq x_d \text{ and } x_j \neq x_d \\ 0 & \text{if } x_i = x_d \text{ and } x_j = x_s \\ \infty & \text{if } x_i = x_d \text{ and } x_j \neq x_s \end{cases}$$

where the costs 0 and ∞ for the edges involving the 'dummy' vertex x_d ensure that the any tour stands for a path that starts in x_s and 'visits' all the atoms in V, while the *distance matrix* $D_s(x_i, x_j)$ encodes the cost of achieving x_j from the state s_i obtained from s by deleting x_s and adding x_i. Such distances can be computed by any delete-based heuristic, including the additive and max heuristics h_a and h_{max} (Bonet & Geffner 2001). When the multi-valued variable X stands for a *root variable* in the causal graph (Helmert 2004), a situation that is not uncommon for 'location' variables in planning benchmarks, the max and additive heuristics yield equivalent distances that are *optimal*. Moreover, in such a case, *it is sufficient to precompute the distances once*, as they do not change when the state s changes. In the general case, such distances need to be recomputed for each state, which is not critical as long as the size of X is not too large. This, however, can be optimized with caching, as the distances $D_s(x_i, x_j)$ do not depend on the whole state s, but only on the values of the variables that are ancestors of X in the causal graph.

Our current implementation only supports the computation of the TSP heuristic for a 'root variable' X in the causal graph, which automatically complies with the assumption above that actions that affect X do not affect any other atoms. The set of helpful actions for the EHC search is changed to be the union of the helpful actions in $\pi(G; s)$ along with the action in the resulting TSP tour that deletes the X-atom true in s (recall that actions affecting X do not appear explicitly in $\pi(G; s)$).

Soft Goals

In many problems, there is a preference over atoms p in a problem expressed by positive rewards $rwd(p)$, so that plans are sought that achieve all the 'hard' goals (if any) while satisfying such preferences as much as possible. We denote that a plan π makes an atom p true as $\pi \models p$. The value $v(\pi)$ of a plan that achieves all hard goals can then be formalized as the sum of the gathered rewards minus the cost of the plan:

$$v(\pi) = \sum_{p: \pi \models p} rwd(p) - \sum_{a_i \in \pi} cost(a_i) \quad (13)$$

so that plans with max value are sought.

Variations of this model of planning with soft goals have been recently considered in (Smith 2004; Sanchez & Kambhampati 2005; Bonet & Geffner 2006). Here we want to show that **soft goals can be compiled away** resulting in a problem with more fluents, actions, and hard goals, but no soft goals. This transformation is important as it implies that any cost-sensitive heuristic can be used to handle soft goals.

Let P be a Strips planning problem extended with cost information $c(a) \geq 0$ over its actions, and reward information $rwd(p) \geq 0$ over a subset of its atoms ($rwd(p)$ for all other atoms is assumed to be zero). For simplicity, we will assume first that atoms p with strictly positive reward $rwd(p) > 0$ cannot be deleted. This is not a central issue in dealing with soft goals, and we will later see how this restriction can be dropped. We will call such soft goals *persistent*.

We define a Strips problem P' with action costs $c(a) \geq 0$ and *no rewards*, such that there is a direct correspondence between the optimal plans for P and P'. Let $S(P)$ stand for the set of 'soft goals' in P:

$$S(P) = \{p \mid p \in F \ \wedge \ rwd(p) > 0\}$$

and let

$$S'(P) = \{p' \mid p' \notin F \ , \ p \in S(P)\}$$

stand for a set of atoms p' not in P, in correspondence with the soft goals p. Then for $P = \langle F, I, G, O \rangle$ where F is the set of fluents, I and G are the initial and goal situations, and O is the set of actions, the problem $P' = \langle F', I', G', O' \rangle$ can be defined as:

- $F' = F \cup S'(P)$
- $I' = I$
- $G' = G \cup S'(P)$
- $O' = O \cup \{Collect(p), Forgo(p) \mid p \in S(P)\}$

where $Collect(p)$ has precondition p, effect p', and cost 0, while $Forgo(p)$ has an empty precondition, effect p' and cost equal to $rwd(p)$.

In words, we 'copy' the soft goals p into atoms p' and make them 'hard goals' in P'. Such hard goals can then be achieved with the action $Collect(p)$ that requires p to be true, at no cost, or can be achieved 'artificially' by means of the action $Forgo(p)$ at the price of not collecting the reward for p. We will use this transformation in the experiments below.

Proposition 2 (Elimination of Soft Goals) *An action sequence π is an optimal plan for the problem P with* persistent soft goals *if and only if π is the result of stripping off the $Collect$ and $Forgo$ actions in an optimal plan for the problem P' with* no soft goals.

[3] Notice that the goal G is a dummy atom not in X.

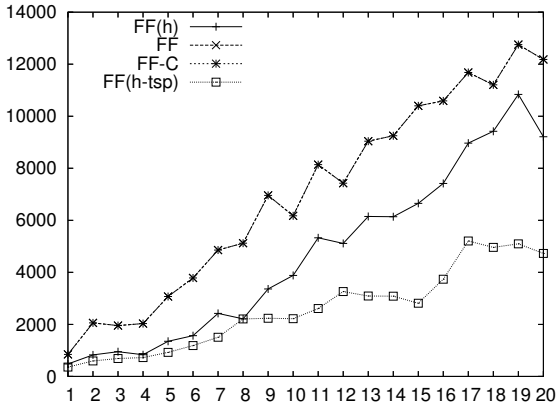

Figure 4: Plan Costs for Rocks with Soft Goals

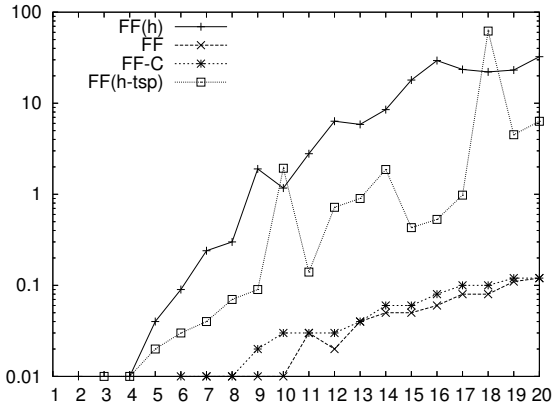

Figure 5: Plan Times for Rocks with Soft Goals

A sketch for the proof proceeds as follows. A plan π is optimal for P if it maximizes the value $v(\pi)$ in (13). The plan π remains optimal as well if we subtract the constant $R = \sum_p rwd(p)$ given by the sum of *all possible rewards* to the plan metric, resulting in a value function $v'(\pi)$ over all plans π given by:

$$v'(\pi) = -\sum_{p:\pi \not\models p} rwd(p) - \sum_{a_i \in \pi} cost(a_i)$$

Maximizing this value function is equivalent in turn to *minimizing* the function

$$c'(\pi) = \sum_{p:\pi \not\models p} rwd(p) + \sum_{a_i \in \pi} cost(a_i)$$

which is exactly the cost of the plan π' obtained in P' by appending to π the $Collect(p)$ actions for the atoms p achieved by π, and the $Forgo(p)$ actions for the atoms p not achieved by π ($\pi \not\models p$).

When soft goals are not persistent, i.e., there are actions in P that may delete them once they are achieved, the transformation of P into P' is slightly different, as it must enforce that all $Collect(p)$ actions occur after the actions that may delete p in P. For *computing* delete-relaxation heuristics, however, this modification is not needed as in the delete-relaxation, all atoms, soft-goals or not, are persistent (the same is true for the TSP heuristic $h_X(s)$ as long as no X-atom is a soft goal).

Evaluating the h_X heuristic

The h_X heuristic is implemented on top of the h^s_a heuristic and uses the 2-opt algorithm for solving the TSP subtasks. The k-opt algorithm, for $k = 2, 3, \ldots$ is a family of local search algorithms for the TSP that try to improve the current tour by swapping k edges in the tour (Lawler & Rinnooy-Kan 1985; Papadimitriou & Steiglitz 1999).

We tested the h_X heuristic with EHC on a rover-type domain similar to the one discussed in (Smith 2004), where rocks at various places must be visited, sampled and their data transmitted. The rocks are placed in grid with size

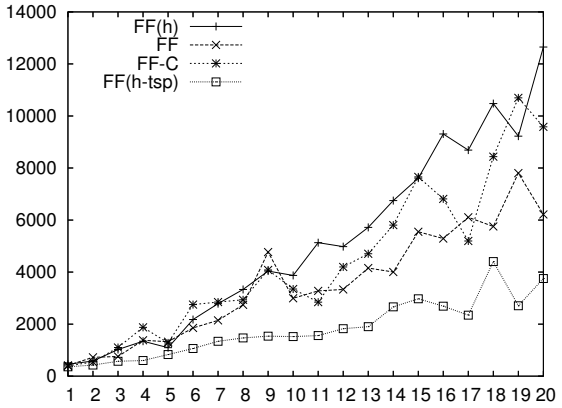

Figure 6: Plan Costs for Rocks with hard goals

100 x 100, with the travel costs between rocks given by the Euclidean distances and 75% of these paths assumed to be traversable. In one version of the problem, the goals are 'soft', i.e. there is a reward for sending data for a rock. In the other the goals are 'hard'. In the soft goals domain, rewards for rocks are selected randomly in the range 0 – 200.

Figures 4 and 6 show the costs of the plans produced by the h_X heuristic, with X being the rover location, in comparison with the heuristics h_{FF}, h^c_{FF}, and h^s_a. The horizontal axis shows the variable number of rocks in the problem set, ranging from 5 to 62. Clearly, the h_X heuristic finds by far the best plans in both variants of the domain, doing very well in time as well. In the soft-version of the problem, FF and FF-C are faster because they simply forgo all the rewards. Surprinsingly, the plans found by the h^s_a heuristic in the hard version of the problem are inferior to both; we have yet to understand why.

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 by modifying

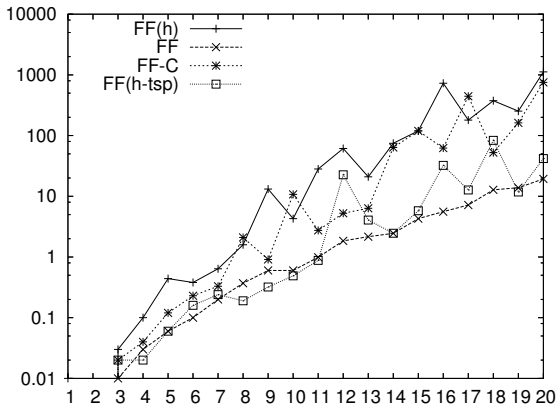

Figure 7: Plan times for Rocks with hard goals

the former to yield relaxed plans that can be used in the EHC search. The resulting formulation suggests other refinements that result from the propagation of labels rather than numbers in the equations. We have considered one such extension, where the required values of a designated multivalued variable X are accumulated and spanned by a path computed by a quick but suboptimal TSP algorithm. The resulting heuristic h_X approximates the cost of the relaxation P^+_X of the problem P where the only deletes that are preserved are the ones involving the atoms x_i associated with the multi-valued variable X, a relaxation that Patrik Haslum has aptly called the *all-but-X delete-relaxation*. We have also shown that soft goals can be compiled away and presented some experimental results. A number of practical issues remain open, such as the automatic selection of the TSP variable, the exploitation of precedence constraints on the possible tours, and the efficient implementation of the approach when the TSP variable is not a root of the causal graph or when multiple TSP variables must be accommodated. Recently, Mirkis and Domshlak have proposed a different variant of the additive heuristic which is based on the propagation of cost vectors rather than sets (Mirkis & Domshlak 2007), whose relation to the ideas proposed in this paper is worth exploring too.

Acknowledgements. We thank B. Bonet and P. Haslum for useful discussions about this topic, and the anonymous reviewers for useful comments, including the suggestion to add the heuristic h^c_{FF} (FF-C) in the evaluation. H.Geffner is partially supported by Grant TIN2006-15387-C03-03 from MEC, Spain.

References

Bertsekas, D. 1991. *Linear Network Optimization: Algorithms and Codes*. MIT Press.

Blum, A., and Furst, M. 1995. Fast planning through planning graph analysis. In *Proceedings of IJCAI-95*, 1636–1642. Morgan Kaufmann.

Bonet, B., and Geffner, H. 2001. Planning as heuristic search. *Artificial Intelligence* 129(1–2):5–33.

Bonet, B., and Geffner, H. 2006. Heuristics for planning with penalties and rewards compiled knowledge. In *Proc. KR-06*.

Brafman, R., and Domshlak, C. 2006. Factored planning: How, when, and when not. In *AAAI-06*.

Cormen, T. H.; Leiserson, C. E.; and Rivest, R. L. 1989. *Introduction to Algorithms*. The MIT Press.

Do, M. B., and Kambhampati, S. 2001. Sapa: A domain-independent heuristic metric temporal planner. In *Proc. ECP 2001*, 82–91.

Fuentetaja, R.; Borrajo, D.; and Linares, C. Improving relaxed planning graph heuristics for metric optimization. In *Proc. 2006 AAAI Workshop on Heuristic Search*.

Helmert, M. 2004. A planning heuristic based on causal graph analysis. In *Proc. ICAPS-04*, 161–170.

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

Hoffmann, J. 2003. The metric-ff planning system: Translating "ignoring delete lists" to numeric state variables. *J. Artif. Intell. Res. (JAIR)* 20:291–341.

Lawler, E., and Rinnooy-Kan, A., eds. 1985. *The Traveling Salesman Problem : A Guided Tour of Combinatorial Optimization*. Wiley.

Long, D., and Fox, M. 2000. Extracting route-planning: First steps in automatic problem decomposition. In *Proc. AIPS Workshop on Analysing and Exploiting Domain Knowledge for Efficient Planning*.

Mirkis, V., and Domshlak, C. 2007. Cost-sharing approximations for h^+. In *Proc. ICAPS-07*.

Papadimitriou, C., and Steiglitz, K. 1999. *Combinatorial Optimization : Algorithms and Complexity*. Dover.

Sanchez, R., and Kambhampati, S. 2005. Planning graph heuristics for selecting objectives in over-subscription planning problems. In *Proc. ICAPS-05*.

Sapena, O., and Onaindia, E. 2004. Handling numeric criteria in relaxed planning graphs. In *Advances in Artificial Intelligence: Proc. IBERAMIA 2004, LNAI 3315*, 114–123. Springer.

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *Proc. ICAPS-04*, 393–401.