Trees of Shortest Paths vs. Steiner Trees: Understanding and Improving Delete Relaxation Heuristics

Emil Keyder
Universitat Pompeu Fabra
08003 Barcelona, SPAIN
emil.keyder@upf.edu

Héctor Geffner
ICREA & Universitat Pompeu Fabra
08003 Barcelona, SPAIN
hector.geffner@upf.edu

Abstract

Heuristic search using heuristics extracted from the delete relaxation is one of the most effective methods in planning. Since finding the optimal solution of the delete relaxation is intractable, various heuristics introduce independence assumptions, the implications of which are not yet fully understood. Here we use concepts from graph theory to show that in problems with unary action preconditions, the delete relaxation is closely related to the Steiner Tree problem, and that the independence assumption for the set of goals results in a tree-of-shortest-paths approximation. We analyze the limitations of this approximation and develop an alternative method for computing relaxed plans that addresses them. The method is used to guide a greedy best-first search, where it is shown to improve plan quality and coverage over several benchmark domains.

1 Introduction

Heuristic search using a non-admissible approximation of the optimal solution to the delete relaxation has proven to be one of the most effective methods in planning. Some of the most informative heuristics for the delete relaxation are the additive heuristic [Bonet and Geffner, 2001], the relaxed plan heuristic [Hoffmann and Nebel, 2001], and a recent variant, the set-additive heuristic, which is based on the recursive computation of relaxed plans [Keyder and Geffner, 2008]. Since solving the delete relaxation optimally is an intractable problem, these heuristics introduce independence assumptions: either that the cost of achieving two goals is the sum of the costs of achieving each, or that the relaxed plan for achieving them is the union of the plans for achieving each. While these assumptions make the problem simple and tractable, their implications are not well understood, though they have a critical impact on the performance of planners.

In order to illustrate the impact of the independence assumption, consider the problem shown in Figure 1, where an agent at L_0 must perform two tasks t_1 and t_2, each available at the locations indicated. Clearly, the best plan is to perform t_2 at L_2 and t_1 at L_3; moreover, the same plan is best in the delete relaxation. However, no current planning heuristic computes such a relaxed plan. Due to the independence assumption embedded in most estimators, what is computed is the union of the plan for getting to L_1 or L_3 to perform t_1 and the plan for getting to L_2 to perform t_2. Since these subgoals are achieved *independently*, the cheaper location L_1 is preferred for t_1 rather than the more expensive location L_3. The result is not just an inaccurate estimate: if the agent can move one unit at a time, it will find that the actions of moving to the left or to the right are both 'helpful', yet that neither results in a state with a lower heuristic value.

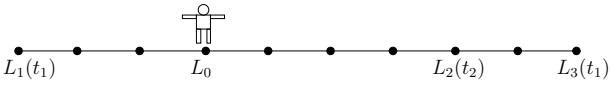

Figure 1: Agent at L_0 must perform tasks t_1 and t_2; t_1 can be done at L_1 or L_3, while t_2 must be done at L_2.

The question that we address here is whether a better estimator for the delete relaxation is possible, in particular, one that is not bound by the independence assumption. Optimally solving the delete relaxation h^+ is intractable [Bylander, 1994], yet as we will show, other options are available. We will draw on concepts from graph theory to better understand the properties of the approximations to h^+ computed by current heuristics, and discuss how these approximations can be improved. We show for example that these heuristics compute the *tree-of-shortest-paths* approximation to the intractable Steiner Tree Problem (STP), closely related to the Minimum Spanning Tree (MST) problem, while h^+ captures the optimal cost. However, a simple criterion for eliminating some non-optimal solutions to the STP can be adapted to the planning setting to give a local iterative improvement algorithm for relaxed plans.

The paper is organized as follows: We first review existing heuristics and consider a planning encoding of the STP. We then use the analysis of this problem to formulate a procedure for improving relaxed plans and test it experimentally over several benchmark domains.

2 Delete-Relaxation Heuristics

We consider planning problems $P = \langle F, I, O, G\rangle$ expressed in STRIPS, where F stands for the set of fluents, $I, G \subseteq F$ the initial and goal situations, and O the actions, with precondition, add, and delete lists $Pre(a)$, $Add(a)$, and $Del(a)$ respectively, all of which are subsets of F. For each action

$a \in O$, we assume a non-negative cost $c(a)$ so that the cost of a plan $\pi = \langle a_1, \ldots, a_n \rangle$ is $c(\pi) = \sum_{i=1}^{n} c(a_i)$. The cost of a problem, denoted by $c^*(P)$, is the cost of the best (minimum cost) plan for P.

The delete relaxation P^+ of P is the same problem but with the delete lists $Del(a)$ assumed empty for all actions. Since computing $c^*(P^+) = h^+$ is intractable, heuristics approximate this value by introducing *independence assumptions*. In the additive heuristic, $h(p; s)$ denotes the estimated cost of achieving a fluent p from s, and is given by:

$$h(p; s) = \begin{cases} 0 & \text{if } p \in s \\ h(a(p; s); s) & \text{otherwise} \end{cases} \quad (1)$$

where $h(a; s)$ stands for an estimate of the cost of applying action a in s, and $a(p; s)$ is a *best support* of fluent p in s:

$$h(a; s) \;=\; cost(a) + \sum_{q \in Pre(a)} h(q; s) \quad (2)$$

$$a(p; s) \;=\; \text{argmin}_{a \in O(p)} h(a; s) \quad (3)$$

where $O(p) = \{a \in O \mid p \in Add(a)\}$. The heuristic estimate of the cost of achieving the goal G from s is

$$h(s) = \sum_{p \in G} h(p; s) \; . \quad (4)$$

Equations 2 and 4 make explicit the independence assumption in the additive heuristic and give it its name: the cost of achieving a set of goal or precondition fluents is estimated as the sum of the costs of achieving each.

A close variant of the additive heuristic is the max heuristic h_{max}, where the summation in equations (2) and (4) is replaced by *max*. h_{max} is admissible but less informed than the additive heuristic [Bonet and Geffner, 2001]. The close connections between these heuristics and the relaxed plan heuristic introduced in FF [Hoffmann and Nebel, 2001] have been discussed recently, and it has been shown how relaxed plans $\pi_{add}(s)$ and $\pi_{max}(s)$ can be obtained from the additive and max heuristics by collecting the best supports $a(p; s)$ captured by (3) recursively backwards from the goal [Keyder and Geffner, 2008]. Such relaxed plans can be used to yield a tighter approximation of h^+ by using the sum $\sum_{a \in \pi(s)} cost(a)$ instead of (4), which avoids counting actions more than once. Moreover, the construction of such relaxed plans is cost-sensitive, and for uniform costs $\pi_{max}(s)$ is equivalent to $\pi_{\text{FF}}(s)$, the relaxed plan computed by FF using the relaxed planning graph. In these formulations, and more explicitly, in the set-additive heuristic where relaxed plans are computed recursively [Keyder and Geffner, 2008], the *independence assumption* takes a slightly different form: rather than taking the cost of achieving a set of subgoals to be the sum of the costs of achieving each one of them, they take the relaxed plan for them to be the union of the relaxed plans for each.

In problems where actions have a *single precondition*, all these heuristics yield *optimal estimates* for all individual *atoms* p in the delete relaxation (in the case of FF, under the assumption of uniform action costs); yet these estimates are *not* optimal for goals containing *multiple atoms*. The common feature of all of these heuristics is that they are 'greedy': they never pay a higher cost to achieve a subgoal even when this could decrease the cost of achieving the set of all subgoals. In the next section, we analyze this problem further by considering a planning problem that encodes a well known problem in graph theory whose optimal cost is captured by h^+.

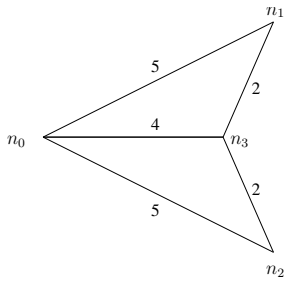

Figure 2: The Steiner Tree for this graph with targets $R = \{n_0, n_1, n_2\}$ has cost 8, which is also the h^+ value of the corresponding planning problem P with $I = n_0$ and $G = \{n_1, n_2\}$. Current heuristics for P yield a relaxed plan with cost 10.

3 Relaxed Plans and Steiner Trees

In order to arrive at a deeper understanding of the weaknesses of current heuristics, we consider a subclass of planning problems with single preconditions and no deletes: those that encode instances of the Steiner Tree Problem (STP). The STP $S = \langle W, R \rangle$ is the problem of finding a minimum cost tree in an undirected graph $W = \langle V, E \rangle$ that spans a set of target nodes $R \subseteq V$, called a Steiner Tree. Here we refer to a tree spanning the set of target nodes R while not necessarily being minimal as a Candidate Steiner Tree (CST). When $R = V$, this is the MST problem, yet the STP is intractable [Proemel and Steger, 2002], while the MST problem can be solved efficiently by greedy algorithms such as Prim's [Cormen *et al.*, 1989]. Prim's algorithm starts with a set N containing initially an arbitrary node in V, and constructs the MST by iteratively selecting a cheapest edge (n, n') in the graph W such that n is in N (spanned already) and n' is not, until $N = V$.

As an illustration, the ST for the graph shown in Figure 2 and the targets $R = \{n_0, n_1, n_2\}$ is the tree composed of the edges (n_0, n_3), (n_3, n_1), and (n_3, n_2) and has cost 8.

The STP can be encoded in a straighforward way as a planning problem $P_S = \langle F, I, O, G \rangle$ without deletes. For this, we define the set of fluents as $F = V$, the initial state as $I = \{n_0\}$ for some $n_0 \in R$ (which therefore must be spanned), the goal as $G = R$, and introduce actions $a_{n,n'}$ and $a_{n',n}$ for each (undirected) edge (n, n') in E with costs $c(a_{n,n'}) = c(n, n')$, where $c(n, n')$ is the cost of the edge (n, n') and $a_{n,n'}$ is an action with precondition n and effect n'. An optimal plan π^* for P_S can then be interpreted as an ST $T_{\pi^*} = \{(n, n') \mid a_{n,n'} \in \pi^*\}$. Denoting the cost of an STP S as $c^*(S)$ we have that:

Theorem 1. *The Steiner Trees for the problem $S = \langle W, R \rangle$ are equivalent to the optimal plans for P_S and $c^*(S) = c^*(P_S) = c^*(P_S^+)$.*

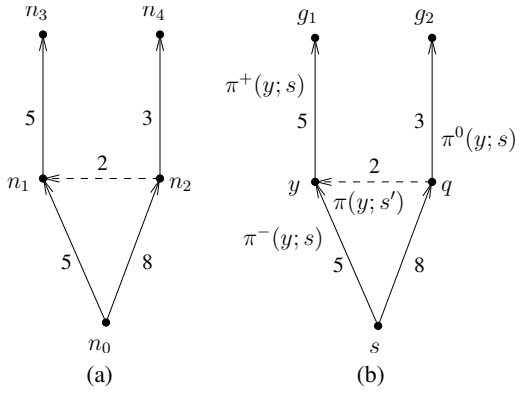

Figure 3: a) Spanning tree on the left, indicated with full lines, can be improved by replacing edge (n_0, n_1) of cost 5 with edge (n_2, n_1) of cost 2. b) Similar iterative improvements can be applied to an arbitrary relaxed plan $\pi(s)$.

While the optimal delete-relaxation heuristic h^+ captures the optimal cost of the resulting planning problem, all of the heuristics considered above compute the *shortest path* to each of the target nodes. The relaxed plans that they yield therefore correspond to *trees of shortest paths* [Cormen *et al.*, 1989]. For example, for the STP shown in Figure 2 with targets $R = \{n_0, n_1, n_2\}$, the heuristics produce a relaxed plan that captures the shortest path from n_0 to n_1 and the shortest path from n_0 to n_2, for a total cost of 10.

The cost of a tree of shortest paths is not a good approximation to the cost of an STP. Yet many polynomial-time algorithms that compute better approximations exist [Charikar *et al.*, 1998; Robins and Zelikovsky, 2000]. We cannot build directly on these algorithms, however, as the graphs that underlie the delete relaxations of arbitrary planning problems are *directed hypergraphs*, rather than undirected graphs. However, as we will show below, it is possible to obtain better relaxed plans by borrowing some of the underlying ideas.

Solving the STP $S = \langle W, R\rangle$ with $W = \langle V, E\rangle$ turns out to be equivalent to the problem of finding a set of *Steiner Points* $Q \subseteq V \setminus R$ such that the cost of the MST over the graph induced in W by $Q \cup R$ is minimized. It is easy to see that this MST is then a solution to S. This suggests a fast test to eliminate a tree T spanning a set of nodes N, $R \subseteq N$, from consideration: if T is not an MST over the graph induced by N, T can be replaced in polynomial time by the MST that spans the same nodes with less cost. Furthermore, these improvements can be done incrementally as follows. Pick an edge $e = (n, n')$ in T and remove it from T. Then check whether there is an edge $e' = (n'', n''')$ such that $c(e') < c(e)$, $n'' \in v(T_1)$, and $n''' \in v(T_2)$, where T_1 and T_2 are the two connected components in $T \setminus e$. If such an edge can be found, then $T_1 \cup T_2 \cup e'$ is a spanning tree of lower cost for the same set of nodes as T. When no further edges can be replaced, it can be shown that the resulting tree is an MST over the set of nodes that it spans. For the graph shown in Figure 3a where $R = \{n_3, n_4\}$, this simple improvement procedure leads us to replace the edge $\langle n_0, n_1\rangle$ with $\langle n_2, n_1\rangle$.

As our overall goal is not to find an MST over the set of nodes spanned by a Candidate Steiner Tree (CST) but rather to approximate the cost of the STP S as closely as possible, we observe that this algorithm can be improved further. In particular, rather than improving the cost of T by replacing a single edge, we can try to replace a *path* $P = \langle n_1, \ldots, n_k\rangle$, where $k \geq 2$, $\langle n_i, n_{i+1}\rangle \in T$ for $1 \leq i \leq k-1$, and $n_i \notin R$ for $1 < i < k$. $T \setminus P$ then also consists of two disjoint connected components T_1 and T_2, with $n_1 \in v(T_1)$ and $n_k \in v(T_2)$. If a path $P' = \langle n'_1, \ldots, n'_l\rangle$ such that $n'_1 \in T_1$, $n'_l \in T_2$, and $c(P') < c(P)$ can be found, a new CST of lower cost results from replacing P with P': $T' = T_1 \cup T_2 \cup P'$. This more powerful improvement procedure generalizes the one above that works only on edges, and may change the set of Steiner points Q of the CST.

The iterative procedure below for improving the cost of relaxed plans can be understood as a generalization of this procedure with relaxed plan fragments playing the role of paths.

4 The Local Steiner Tree (LST) Improvement Procedure

The graphs induced by the delete relaxations of most planning problems differ from the undirected graphs discussed until now in two important respects. First, the delete-relaxation graph is *directed*, its edges (actions) having source nodes (preconditions) and target nodes (add effects). Second, it is actually a *hypergraph*, as both precondition and add sets may have cardinality greater than 1. Solving the delete relaxation optimally is therefore the problem of optimal directed hyperpath finding [Ausiello *et al.*, 1992]. Here we develop a technique inspired by the idea of the delete relaxation as an STP, yet applicable in this more general setting.

In what follows, we assume the presence of a dummy *End* action with *zero cost*, whose preconditions $G_1, \ldots, G_n$ are the goals of the problem, and whose effect is a new dummy atom G. This allows us to express the relaxed plan from the state s, $\pi(s)$, as the relaxed plan for G, $\pi(G; s)$. We interpret relaxed plans as sets of action-fluent pairs $\langle a, p\rangle$ where a is the best supporter of p in s, i.e. $a(p; s) = a$. Relaxed plans are computed by recursively collecting the atoms and their best supporters backwards from the goal G, stopping the recursion at atoms appearing in s. Below we abuse notation slightly by allowing relaxed plans to represent both this set of pairs and the set of actions appearing in them; the corresponding set of fluents is denoted by $F(\pi)$.

To improve the cost of a relaxed plan $\pi(s) = \pi(G; s)$, we pick a fluent $y \in F(\pi(s))$ and split the set of actions $\pi(s)$ into three disjoint subsets. $\pi^-(y; s)$ is the set of actions that are required *only* in order to achieve y, and corresponds to the path P to be replaced in the algorithm discussed earlier. $\pi^+(y; s)$ is the set of actions that depend on y being made true in order to be applied, and corresponds to one of the connected components that results from removing a path from a tree. Finally, $\pi^0(y; s)$ is the set of actions that are neither used exclusively in order to achieve y nor depend on its being made true, and corresponds to the other connected component. We then check whether it is possible to improve the part of the plan that is exclusively for y, $\pi^-(y; s)$ *in the context of the rest of the plan*. This is accomplished by computing a re-

laxed plan $\pi(y; s')$ for y in a new state s' that extends s with atoms that are supported by $\pi^0(y)$ while leaving unchanged the best supports in $\pi^+(y; s)$ and $\pi^0(y; s)$. As before, if the cost $c(\pi(y; s'))$ is lower than $c(\pi^-(y; s))$, then a better relaxed plan for $\pi(G; s)$ results from replacing $\pi^-(y; s)$ with $\pi(y; s')$. The idea is shown in Fig. 3b along with the graph where the same idea was used to improve a Candidate Steiner Tree.

For a relaxed plan $\pi(G; s)$ and a fluent $y \in F(\pi(s))$, the three sets above can be defined as follows:

$$\begin{aligned}
\pi^-(y; s) &= \{a \in \pi(y; s) \mid a \notin \cup_{z \in F(\pi(s)) \setminus F(\pi(y;s))} \pi(z; s)\} \\
\pi^+(y; s) &= \{a \in \pi(G; s) \mid y \in \cup_{p \in Pre(a)} F(\pi(p; s))\} \\
\pi^0(y; s) &= \pi(G; s) \setminus (\pi^+(y; s) \cup \pi^-(y; s))
\end{aligned}$$

To compute $\pi^-(y; s)$, we first extract a plan for G in s that *assumes* the fluent y, denoted $\pi(G; s|y)$. This is done with a simple modification to the relaxed plan extraction procedure that returns the empty plan for y whenever it is encountered as a precondition. We then have that $\pi^-(y; s) = \pi(G; s) \setminus \pi(G; s|y)$. $\pi^+(y; s)$ is obtained by calling a recursive procedure on the goal fluent G that determines the portion of the relaxed plan for a fluent that is dependent on y. $\pi^0(y; s)$ is computed in the manner implied by its definition.

The new relaxed plan $\pi(y; s')$ for y is obtained from an extension s' of the state s that contains all fluents $\{p \mid a(p; s) \in \pi^0(y; s)\}$. The improved plan $\pi'(G; s)$ is then given by the union of $\pi^0(y; s)$ and $\pi^+(y; s)$ with the new fragment $\pi(y; s')$ that replaces $\pi^-(y; s)$. This new relaxed plan is well formed and has no cycles, as the preconditions of all actions that directly or indirectly use y (those in $\pi^+(y; s)$) are excluded in the computation of the new relaxed plan $\pi(y; s')$ for y.

The pseudocode for a single iteration of the resulting method for improving the cost of relaxed plans is shown in Algorithm 1. The improvement algorithm is repeatedly called until it returns false, at which point the improved relaxed plan $\pi_{lst}(s)$ is obtained in the usual way by recursion backwards from the goal G collecting the best supports.

Algorithm 1 LST Procedure for improving relaxed plan $\pi(s)$.

```
for y ∈ F(π(G; s)) do
  Compute π⁻(y; s), π⁺(y; s), π⁰(y; s)
  s' = s ∪ {p | a(p; s) ∈ π⁰(y; s)}
  D = {p | a(p; s) ∈ π⁺(y; s)}
  Compute π(y; s') with Add(a) := Add(a) \ D for all a
  if Cost(π(y; s')) < Cost(π⁻(y; s)) then
    for q ∈ F(π(y; s')) do
      a(q; s) = a(q; s')
    end for
    return true
  end if
end for
return false
```

5 Example

Here we illustrate the functioning of the relaxed plan improvement algorithm on a problem involving soft goals [Smith, 2004; Sanchez and Kambhampati, 2005]. It has been shown that soft goals p with utility $u(p)$ that no action deletes can be compiled away by the introduction of extra fluents p' that become normal (hard) goals of the problem, and two actions *forgo*(p) and *collect*(p) that add p': the first with no preconditions and cost $u(p)$, the second with precondition p and cost 0 [Keyder and Geffner, 2007].[1] We consider an example with soft goals as previous greedy heuristics tend to do poorly in this context.

The problem is shown in Figure 4. The agent begins at location A, and there are two packages available to gather from the map, each with an associated reward as shown. The relaxed plan $\pi(s)$ generated by the additive heuristic for the initial state $s = \{(at\ A)\}$ is:

$$\{(A \rightarrow B), (B \rightarrow C), (\textit{pick}\ p_1), (\textit{coll}\ p_1), (\textit{forgo}\ p_2)\}$$

where the *End* action is omitted. This relaxed plan achieves the two goals, p'_1 and p'_2, with the cheapest plan for each when considered independently. Yet since the agent will move to locations B and C in order to pick up p_1, it also makes sense to pick up p_2 as the additional cost incurred to reach location D is 2, and there is a soft goal that is achievable there with reward 3. The set $F(\pi(s))$ generated by the above plan is

$$\{(at\ B), (at\ C), (\textit{have}\ p_1), (p'_1), (p'_2)\}\ .$$

The only fluent in this set that can be improved with the LST procedure is p'_2. The subsets of $\pi(s)$ then become

$$\begin{aligned}
\pi^+(p'_2; s) &= \{\textit{End}\} \\
\pi^-(p'_2; s) &= \{\textit{forgo}\ p'_2\} \\
\pi^0(p'_2; s) &= \{(A \rightarrow B), (B \rightarrow C), (\textit{pick}\ p_1), (\textit{coll}\ p_1)\}
\end{aligned}$$

This gives $s' = \{(at\ A), (at\ B), (at\ C), (\textit{have}\ p_1), (p'_1)\}$, from which we compute $\pi(p'_2; s')$:

$$\{(C \rightarrow D), (\textit{pick}\ p_2), (\textit{coll}\ p_2)\}$$

which has a cost of 2 compared to the cost of $\pi^-(p'_2; s)$ which was 3. The supporters for the relevant fluents are then replaced, resulting in the new relaxed plan with cost 6, compared to the cost of the original relaxed plan, 7:

$$\begin{aligned}
\pi'(s) = \{&(A \rightarrow B), (B \rightarrow C), (C \rightarrow D), \\
&(\textit{pick}\ p_1), (\textit{coll}\ p_1), (\textit{pick}\ p_2), (\textit{coll}\ p_2)\}
\end{aligned}$$

No further improvements are possible at this point, so the procedure terminates, returning the new relaxed plan.

6 Experimental Results

We evaluated the performance of two heuristics on ten different domains. Nine of these domains were from the most recent international planning competition (IPC6). The tenth domain is a compiled soft goals domain [Keyder and Geffner, 2007], similar to the Rovers domain from IPC3.

[1] A slightly different compilation is required when soft goals can be deleted, for details, see [Keyder and Geffner, 2007].

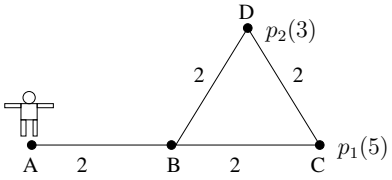

Figure 4: A softgoals problem.

The two heuristics used were the costs of the relaxed plan extracted according to the best supporters as determined by the additive heuristic, and the cost of the same plan as improved by the LST procedure. In the following graphs, these two heuristics are denoted by h_a and h_{lst} respectively. The search algorithm used was greedy best-first search with delayed evaluation and two open lists, one containing only states arrived at through helpful actions [Helmert, 2006]. The helpful actions used were those adding at least one precondition of some action in the relaxed plan. The planners were evaluated using the same time and memory settings as IPC6, with a timeout of 1800 seconds and a memory limit of 2GB. The experiments were run on Xeon Woodcrest computers with clock speeds of 2.33 GHz.

The use of the LST procedure significantly impacts results in three domains: parcprinter, elevators, and softgoals. In all of these domains, the number of nodes expanded decreases by orders of magnitude (Figures 6a,6b,6c). In the elevators and softgoals domains, plan cost also decreases significantly (Figures 5a, 5b). In the parcprinter domain, solutions found by both heuristics have equal cost, yet h_{lst} increases coverage from 16 to 24 problems out of 30 due to the much lower number of nodes evaluated to find a solution.

In domains in which no improvement is observed, the LST procedure decreases coverage slightly due to the increased overhead, and the number of nodes evaluated and solution cost are roughly equal for all problems. Total coverage across all domains studied, however, is higher with the LST procedure than for the base heuristic h_a (Table 1).

There is a wide variation in the correlation between improvement to relaxed plan cost and improvement to overall plan cost. While the correlation is high in the domains in which the greatest improvement to plan quality is observed, the greatest improvement to relaxed plan cost occurs in the pegsol domain, in which the LST procedure leads to no appreciable change in overall plan quality (Table 1).

Domain	h_a	h_{lst}	(a)	(b)
cybersec (30)	11	9	1.00	0.99
elevators (30)	11	19	0.72	0.79
openstacks (30)	30	30	1.00	1.00
parcprinter (30)	16	24	1.00	0.81
pegsol (30)	30	29	0.99	0.64
scanalyzer (30)	22	18	0.98	0.92
sokoban (30)	27	23	1.04	0.93
transport (30)	14	15	0.95	0.85
woodworking (30)	27	28	0.98	0.90
softgoals (30)	26	28	0.65	0.66
total/avg.	214	223	0.93	0.85

Table 1: The first two columns show coverage for tested domains. (a) Average ratio of the *cost of the plan* found with h_{lst} to the plan found with h_a, for problems solved with both heuristics. (b) Average ratio of the *cost of the relaxed plan* improved with the LST procedure to the cost of the original h_a relaxed plan (for all evaluated nodes).

7 Summary

Current heuristic estimators in planning approximate the optimal cost h^+ of the delete relaxation by introducing independence assumptions over either estimated costs or relaxed plans. This assumption results in a 'greedy' estimator that attempts to minimize the cost of achieving individual fluents rather than sets. We have analyzed this problem by considering the Steiner Tree problem, where current heuristics yield the tree-of-shortest-paths approximation, and have developed a method for improving relaxed plans inspired by its properties. The results obtained from the resulting heuristic over several domains are encouraging and suggest that less greedy approximations to the optimal but intractable h^+ heuristic may be both desirable and cost-effective.

Acknowledgments

We thank the anonymous reviewers for their helpful comments. H. Geffner is partially supported by grant TIN2006-15387-C03-03 from MEC/Spain.

References

[Ausiello *et al.*, 1992] G. Ausiello, R. Giaccio, G. F. Italiano, and U. Nanni. Optimal traversal of directed hypergraphs. Technical Report TR-92-073, International Computer Science Institute, 1992.

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

[Bylander, 1994] T. Bylander. The computational complexity of STRIPS planning. *Artificial Intelligence*, 69:165–204, 1994.

[Charikar *et al.*, 1998] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation algorithms for directed Steiner problems. In *Journal of Algorithms*, pages 73–91, 1998.

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

[Helmert, 2006] M. Helmert. The Fast Downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246, 2006.

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

[Keyder and Geffner, 2007] E. Keyder and H. Geffner. Set-additive and TSP heuristics for planning with action costs

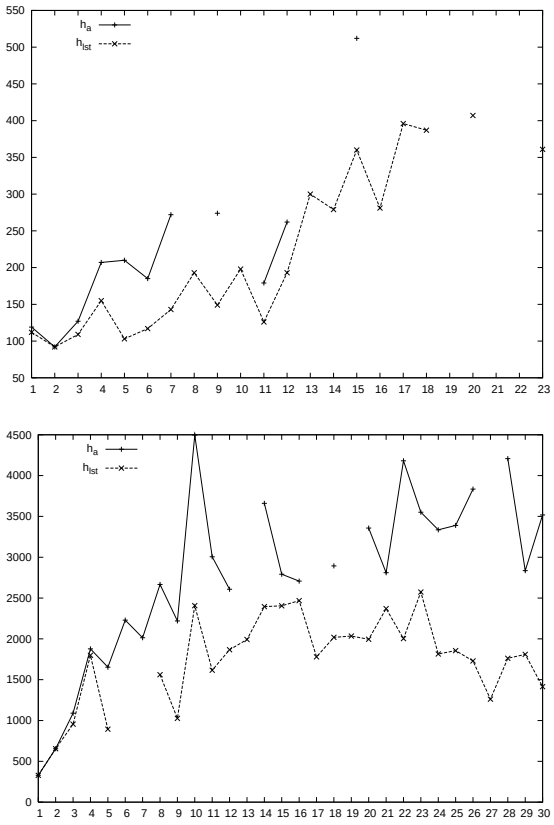

Figure 5: Plan costs in the elevators and softgoals domains.

and soft goals. In *Proc. 2007 ICAPS Workshop on Heuristics for Domain-Independent Planning*, 2007.

[Keyder and Geffner, 2008] E. Keyder and H. Geffner. Heuristics for planning with action costs revisited. In *Proc. 18th European Conference on Artificial Intelligence*, pages 588–592, 2008.

[Proemel and Steger, 2002] H. Proemel and A. Steger. *The Steiner Tree Problem: A Tour Through Graphs, Algorithms, and Complexity*. Vieweg+Teubner Verlag, 2002.

[Robins and Zelikovsky, 2000] G. Robins and A. Zelikovsky. Improved Steiner tree approximation in graphs. In *SODA*, pages 770–779, 2000.

[Sanchez and Kambhampati, 2005] R. Sanchez and S. Kambhampati. Planning graph heuristics for selecting objectives in over-subscription planning problems. In *Proc. ICAPS-05*, pages 192–201, 2005.

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

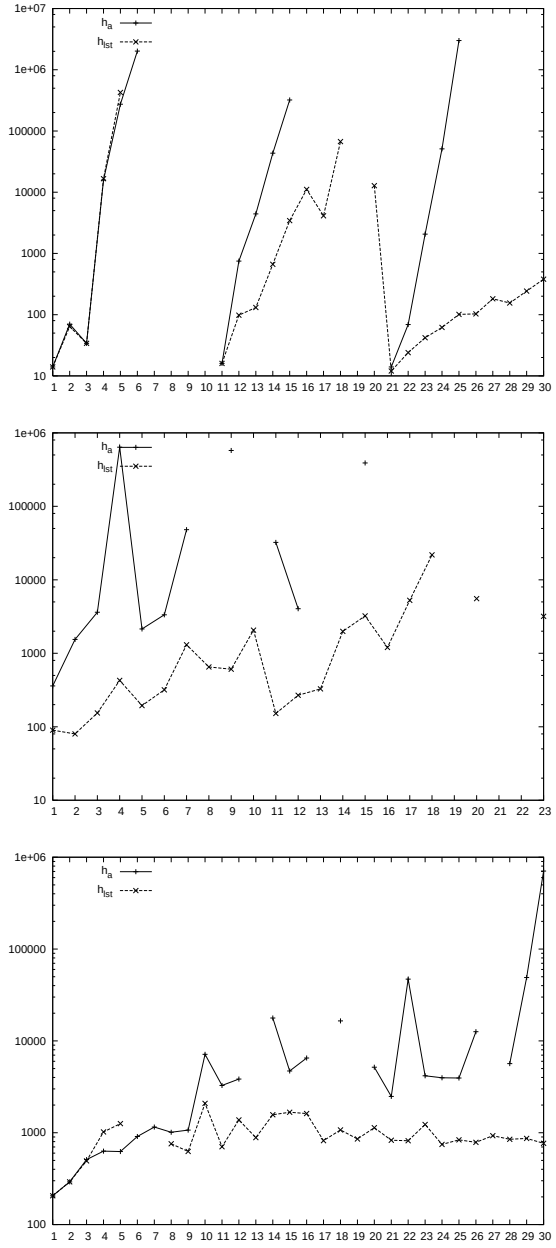

Figure 6: Number of nodes evaluated in the parcprinter, elevators, and softgoals domains.