Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling

Searching for Good Solutions in Goal-Dense Search Spaces

Amanda Coles and **Andrew Coles**
Department of Informatics,
King's College London,
London, WC2R 2LS, UK
email: {*amanda,andrew*}.*coles@kcl.ac.uk*

Abstract

In this paper we explore the challenges surrounding searching effectively in problems with preferences. These problems are characterized by a relative abundance of goal states: at one extreme, if every goal is soft, every state is a goal state. We present techniques for planning in such search spaces, managing the sometimes-conflicting aims of intensifying search around states on the open list that are heuristically close to new, better goal states; and ensuring search is sufficiently diverse to find new low-cost areas of the search space, avoiding local minima. Our approach uses a novel cost-bound-sensitive heuristic, based on finding several heuristic distance-to-go estimates in each state, each satisfying a different subset of preferences. We present results comparing our new techniques to the current state-of-the-art and demonstrating their effectiveness on a wide range of problems from recent International Planning Competitions.

1 Introduction

AI planning has traditionally been concerned with achieving a fixed set of specified goals. More recently following work by Smith (2004) and others, and the subsequent introduction of PDDL3 in the 2006 International Planning Competition (IPC2006), planning problems with preferences (soft constraints) have been considered. Adding more preferences to a planning problem than can be achieved, with corresponding violation costs, creates an over-subscription problem in which the planner must decide which combination of preferences to satisfy in order to find solutions of high quality.

Planning with preferences poses an important challenge. Traditionally, planners use heuristic estimates of cost-to-go or distance-to-go (number of actions), from each state, to a goal state. In planning with preferences, however, goal states are much more abundant; indeed in the case where there are no hard goals, all states are goal states, and the distance-to-go value is always zero. The planner therefore not only needs guidance to reach a goal state but also to reach a *good* goal state that satisfies as many preferences as possible. The current state-of-the-art in (non-temporal) planning with preferences is LPRPG-P (Coles and Coles 2011), where the distance-to-go is based on satisfying all reachable preferences. This provides good overall guidance to reaching as many preferences as possible; but fails to promote the expansion of states that are just a few steps from a new goal state, that is better than the best solution found so far. As argued in the context of search based on cost-to-go, such states should not be ignored in favour of those that lead to goal states that are better, but much further away (Cushing et al. 2011).

In this paper we transform the notion of a heuristic value when planning with preferences. Instead of maintaining just a single heuristic distance-to-go, corresponding to reaching all preferences; we generate several heuristic values for each state, corresponding to the length of relaxed plans (Hoffmann and Nebel 2001) for achieving different subsets of the reachable preferences. Each relaxed-plan length value is paired with a cost-to-go value, reflecting the cost of violating the excluded preferences. During search, these length–cost pairs underpin a cost-bound-sensitive distance-to-go heuristic: as the upper-bound on acceptable solution cost is tightened, the requisite relaxed plan length is increased, to reflect the higher quality now demanded.

We evaluate the use of this new heuristic alongside the prior LPRPG-P heuristic, using it in dual open-list alternation search (Helmert 2006). The resulting planner is tested on a range of domains, including all those relevant from the International Planning Competition series: the Preferences domains from 2006, and the Net Benefit domains from 2008. Our results show that our approach improves upon the current state-of-the-art in planning with preferences.

2 Background

In classical planning problems, the task is to find a sequence of actions that, when executed, transform the initial state into one in which some goals have been met: a goal state. An important issue that arises is characterisation of good quality plans: not all possible solutions to a problem are equal.

Within PDDL, the first steps to capturing plan costs were made in version 2.1 (Fox and Long 2003). Here, a plan quality metric can be specified, with terms comprising the values of numeric task variables, as measured at the end of the plan. A convention, later standardised in PDDL 3.1, is to use a variable `(total-cost)` to capture action costs, with each action a incrementing this by some non-negative amount $cost(a)$.

Motivated by work such as that of Smith (2004), the notion of preferences was introduced into PDDL version 3.0 (Gerevini et al. 2009). These range from soft goals, or

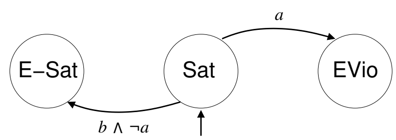

Figure 1: Automaton for `(sometime-before a b)`

precondition preferences on actions; to expressions over the plan trajectory, written in a language capturing a subset of linear temporal logic (LTL). In the case of soft goals, one option is to use a compilation approach (Keyder and Geffner 2009), with dummy actions that collect or forgo each preference; with the latter carrying a cost. This compiled domain can then be used with any planner capable of finding cost-effective plans in terms of action costs, such as Lama (Richter and Westphal 2010).

For the broader class of preferences supported in PDDL3, a range of domain-independent techniques have been developed for reasoning with preferences directly (Edelkamp et al. 2006; Baier et al. 2007; Benton et al. 2009; Coles and Coles 2011; Benton et al. 2012). These tailor their search and heuristics to planning with preferences, in a variety of ways. A common element used by many of these is a Finite State Automaton model of preferences, where each planning state records the current automata positions, and the update of these is synchronised to the application of new actions: each time a new state is reached, the automata are updated accordingly. For instance, Figure 1 shows the automaton for `(sometime-before a b)`. It begins in the 'Sat' position, and if a plan reaches `a` before earlier having reached `b`, the preference is eternally violated: it moves to 'EVio'. This increases the cost of the plan by the defined violation cost of the preference, $cost(p)$ for each preference p, specified in problem as part of the PDDL metric function.

Beyond work on preferences, there is a rich body of work on searching for good-quality solutions, combining heuristic measures of distance-to-go and cost-to-go, from each state to a goal state. One possibility, used in Lama in the last International Planning Competition (Richter, Westphal, and Helmert 2011; Richter and Westphal 2010) is to run a number of searches in succession: first, guided by distance-to-go; and then by cost-to-go, using Restarting WA* (Richter, Thayer, and Ruml 2010). Explicit Estimation Search (EES) (Thayer and Ruml 2011) and its various extensions (Thayer et al. 2012; Thayer, Benton, and Helmert 2012) integrate distance and cost measures more closely: starting with WA* (Pohl 1973) guided by cost-to-go, they add a *focal list* of 'good enough' nodes, sorting this focal list by distance-to-go.

3 Relaxed Planning to Goal States

When extracting a relaxed plan to satisfy hard goals, all the actions chosen are justifiable: they are added to the relaxed plan to satisfy a hard goal, or precondition of some other action in the relaxed plan. If we extend the relaxed planning graph and relaxed-plan extraction to also include preferences, the actions are still justifiable — they satisfy a hard goal, precondition, or preference — but are not necessarily essential. That is, if we remove from the relaxed plan actions which serve *only* to support preferences, the relaxed plan will still lead to a goal state (in the relaxed problem); and with fewer actions, too.

This observation underlies a fundamental trade-off when determining the heuristic value (relaxed plan length) when planning with preferences. If we disregard preferences, and obtain a relaxed-plan only for hard goals, we promote states that are close to achieving the hard goals; but the heuristic value reflects only the effort to reach a poor-quality goal state, in terms of preference violations. Worse, in problems with no hard goals, the heuristic value is universally zero. On the other hand, if we find a relaxed plan to reach the hard goals and as many preferences as possible (under the delete relaxation), we promote states that are close to achieving all of these; but the heuristic value may be far larger than the number of actions necessary to reach the hard goals alone.

To address these issues, we propose producing multiple heuristic values for each state: relaxed-plan length–cost pairs, where higher lengths are paired with lower costs. Core to our method for finding these is a relaxed plan where each action is *annotated* with the reason it was added to the plan. We first recap the production of a relaxed planning graph for preferences, due to Coles & Coles (2011), upon which we build, before introducing our novel technique for the extraction of an annotated relaxed plan. We go on to present a number of ways in which length–cost pairs can be obtained from such a relaxed plan. Our approach supports all PDDL 3 non-temporal preferences with ADL conditions.

3.1 A Relaxed Planning Graph for Preferences

Starting with the classical case, a relaxed planning graph (RPG) consists of alternate fact and action layers, where:

- $fl(0)$, i.e. fact layer 0, is the state being evaluated;
- $al(i)$, i.e. action layer i, contains actions whose preconditions are true in $fl(i\text{-}1)$;
- $fl(i)$ contains all the facts in $fl(i\text{-}1)$, plus those added by any action in $al(i)$.

The RPG can be constructed, iteratively, by beginning with $fl(0)$ and adding alternate action- and fact-layers. Graph expansion terminates at either the first fact layer containing the goals; or if the fix point is reached (no new facts and hence actions are appearing), in which case the goals are unreachable, and the state being evaluated is a dead-end.

When adding preferences to the RPG, we have the additional consideration of tracking their status as the RPG is expanded. As discussed in Section 2, each preference can be represented as a finite-state automaton, with labelled transitions. The appropriate treatment for these depends on the automaton. The key details for our purposes are that:

- For `(sometime F)` or soft goals `(F)`, it is desirable to make F true. The termination criterion for graph expansion is therefore modified to only terminate sooner than the fix point if all such preferences are satisfied.

- In addition to their preconditions as prescribed in the problem description, actions acquire 'soft' preconditions to capture their interaction with preferences, with $soft\_pre(a,t)$ containing the soft preconditions attached to action a in action layer t, as fact–preference pairs. These arise from either:
  - Precondition preferences. For a precondition preference p with formula (F), if $fl(t\text{-}1)$ satisfies F, then for each fact $f \in F$, $(f,p) \in soft\_pre(a,t)$.
  - The interaction with `(sometime-before F′ F)` preferences (where F and F' are logical formulæ). For such a preference p, if F has not yet been reached in the plan, and the effects of a would satisfy F' regardless of the state in which a was executed, then a risks violating the preference. To account for this, if $fl(t\text{-}1)$ satisfies F, then for each fact $f \in F$, $(f,p) \in soft\_pre(a,t)$. This ensures F can be made true prior to the execution of a, and hence making F' true is harmless.
- Each fact f is associated with a (possibly empty) set of preferences that would be broken by the actions applied to achieve the fact. These violations arise either directly (an action achieves f but violates some preference), or indirectly (achieving the preconditions of actions achieving f necessitate violating some preference). The cost of such a set of preferences is taken to be the sum of the violation costs of its constituent preferences.
- As the RPG is expanded, these violation sets monotonically reduce in cost: alternative achievers for facts become available, and in the case of preferences such as `(sometime-before F′ F)`, some actions' effects no longer violate preferences. Thus, the cost of a fact is layer-dependent. For a relaxed planning graph R, and time-bound t', we use $low\_cost\_before(R,f,t')$ to note the latest (cheapest) fact layer $fl(t)$ where $f \in fl(t)$ and $t < t'$.

3.2 Extracting an Annotated Relaxed Plan

Having covered relaxed planning graph construction we can now move on to our first contribution, and the first step towards multiple relaxed plan extraction: a method for extract annotated relaxed plans. As in the classical case, this is a layer-wise process, regressing backwards from the goals. The difference in our case is that we annotate the actions with the reason they were added to the relaxed plan. These annotations take the form of sets, containing preferences, and/or a dummy preference $\mathbf{H}$ to denote hard goals. From hereon, refer to such a set as an annotation. To track annotations during relaxed solution extraction, we employ a queue Q, where $Q[t]$ contains the goals to be satisfied in $fl(t)$. If $Q[t][g]$ is defined (i.e. g has been added as a goal at $fl(t)$) then $Q[t][g]$ is an annotation, attached to g, explaining its presence as an intermediate goal.

Algorithm 1 shows the extraction algorithm. From lines 2 to 10, we seed Q with the hard goals, and any goals introduced by preferences (such as the soft-goals or `sometime` preferences discussed earlier). Facts due to hard goals are annotated with $\mathbf{H}$, and facts due to preferences with the identifier of the relevant preference p. The function

Algorithm 1: Annotated Relaxed Plan Extraction

Data: G: task goals, R: a relaxed planning graph
Result: $\Pi = \{\langle a, s\rangle\}$: a relaxed plan

1 $Q \leftarrow [], \Pi \leftarrow []$;
2 **foreach** $g \in G$ **do**
3     $t \leftarrow low\_cost\_before(R, g, \infty)$;
4     $Q[t][g] \leftarrow \{\mathbf{H}\}$;
5 **foreach** $p \in prefs$ **do**
6     $G' \leftarrow$ top-level goals in R to satisfy p;
7   **foreach** $g \in G'$ **do**
8         $t \leftarrow low\_cost\_before(R, g, \infty)$;
9   **if** *defined* $Q[t][g]$ **then** $Q[t][g] \leftarrow Q[t][g] \cup \{p\}$;
10   **else** $Q[t][g] \leftarrow \{p\}$;
11 $last \leftarrow$ last valid index of Q;
12 **foreach** $t \in [last \ldots 1]$ **do**
13   **foreach** $g \in Q[t]$ **do**
14         $a \leftarrow$ lowest-cost supporter of g in R, layer t-1;
15   add $\langle a, Q[t][g]\rangle$ to front of Π;
16   **foreach** $f \in pre(a)$ **do**
17             $t' \leftarrow low\_cost\_before(R, f, t\text{-}1)$;
18   **if** *defined* $Q[t'][f]$ **then**
19                 $Q[t'][f] \leftarrow Q[t'][f] \cup Q[t][g]$;
20   **else** $Q[t'][f] \leftarrow Q[t][g]$;
21   **foreach** $(f,p) \in soft\_pre(a, t-1)$ **do**
22             $t' \leftarrow low\_cost\_before(R, f, t\text{-}1)$;
23   **if** *defined* $Q[t'][f]$ **then**
24                 $Q[t'][f] \leftarrow Q[t'][f] \cup Q[t][f] \cup \{p\}$;
25   **else** $Q[t'][f] \leftarrow Q[t][f] \cup \{p\}$;
26 **return** Π

low_cost_before is used to choose the layer in which to insert these as goals: by passing ∞ as the time upper-bound, the goal can appear as late (hence as cheaply) as possible.

Once the goals have been added to Q, solution extraction works backwards through the planning graph, choosing actions to meet the goals at each layer. The relevant loop is from line 12 to line 25. In each layer, starting with the last, it loops over the goals to be achieved, i.e. each $g \in Q[t]$. At line 14, an action that supports (adds) g is chosen from layer t-1. This is added to the relaxed plan, paired with the annotations $Q[t][g]$. This ensures that the reason for the action's presence in the relaxed plan is noted.

Each of the preconditions, and any soft preconditions (due to preferences), are then added as goals to be achieved at earlier layers in the planning graph. Note that:

- $low\_cost\_before(R, f, t\text{-}1)$ is used to choose the fact layer at which to request f. As this has to be prior to $al(t\text{-}1)$, this has to be in fact layer $fl(t\text{-}1)$, or earlier.
- The annotations attached to $Q[t'][f]$ are cumulative: it inherits any annotations from $Q[t][g]$, and – for soft preconditions – the relevant preference p.

3.3 Deriving Length–Cost Pairs from Annotated Relaxed Plans

With an annotated relaxed plan, we can now explore the trade-off between the length of the relaxed plan, and the cost of the goal state it reaches. Intuitively, from a relaxed plan with actions to meet all the preferences, we can derive several alternative relaxed plans by keeping just the actions that are relevant to a subset of the preferences. As long as the actions that are relevant to the hard goals remain, it will still be a relaxed plan, though one that satisfies fewer preferences.

As a baseline for this process, we partition the steps of the relaxed plan into those that are due to hard goals, and those that are relevant only to preferences. For this, we use make use of the annotation set s that accompanies each action a in the relaxed plan generated by Algorithm 1. The 'hard' and 'soft' steps in a relaxed plan Π are:

$$hard(\Pi) = [(a, s) \mid (a, s) \in \Pi \wedge \mathbf{H} \in s]$$
$$soft(\Pi) = [(a, s) \mid (a, s) \in \Pi \wedge \mathbf{H} \notin s]$$

Further, the set of preferences P that led to actions being introduced into a relaxed plan Π are:

$$P(\Pi) = \{p \mid \exists (a, s) \in \Pi \text{ s.t. } p \in s\}$$

...and the steps in a relaxed plan Π relevant to some preference p are:

$$relevant(\Pi, p) = [(a, s) \mid (a, s) \in \Pi \wedge p \in s]$$

Assuming all $relevant(\Pi, p)$ must be included to satisfy p, then we can derive a relaxed plan Π' from Π in a two-stage process:

- Choose a subset of preferences $P' \subseteq P(soft(\Pi))$;
- Construct a relaxed plan Π' containing the steps relevant to any $p \in P'$, alongside those for the hard goals:

$$\Pi' = hard(\Pi) \cup (\bigcup_{p \in P'} relevant(soft(\Pi), p))$$

From Π' we have a heuristic estimate of the length of the relaxed plan needed to reach the hard goals and the preferences P'. The key questions are now: what is the heuristic *cost* associated with this? And how do we choose the subset P'?

For cost, we have two options. At the very least, we need to account for the preferences whose actions have not been included in the relaxed plan Π':

$$omit(\Pi', \Pi) = \{p \in P(\Pi) \mid relevant(soft(\Pi), p) \nsubseteq \Pi'\}$$

Taking the sum of the violation costs of these preferences yields a heuristic cost estimate:

$$pcost(\Pi', \Pi) = \sum_{p \in omit(\Pi', \Pi)} cost(p)$$

Optionally, we also may wish to add to this the cost of the actions in the relaxed plan itself, reflecting the net-benefit of adding the actions to achieve the preferences (i.e. to minimise action cost plus preference violation cost). This is not an admissible measure (Do and Kambhampati 2003), and the implementation we use only excludes actions from the RPG if they are trivially too expensive to apply (Coles et al. 2011). Nonetheless, it does incorporate some information about action costs – rather than none at all – so has the potential to be informative:

$$nbcost(\Pi', \Pi) = pcost(omit(\Pi', \Pi)) + \sum_{(a,s) \in \Pi'} cost(a)$$

Taking *cost* to be whichever cost function is chosen from these two options, the length–cost pair for Π' is:

$$\langle |\Pi'|, cost(\Pi', \Pi) \rangle$$

Choosing subsets P' of $P(soft(\Pi))$ to produce these length–cost pairs is more difficult. A naïve approach is to consider all possible P'. The number of possible P' is exponential in the size of $P(soft(\Pi))$[1], so this is unlikely to scale; but it is of theoretical interest as it provides the Pareto front of length–cost pairs that can be obtained from Π using our approach. We say $\langle l, c \rangle$ dominates $\langle l', c' \rangle$ if:

$$dominates(\langle l, c \rangle, \langle l', c' \rangle) = (l \leq l' \wedge c < c')$$

For practical reasons, we focus on length–cost set approximation algorithms, that consider only a small number of P'. Our greedy algorithm for this is shown in Algorithm 2, and at worst it is quadratic (rather than exponential) in the size of $P(soft(\Pi))$. Beginning with a relaxed plan Π' containing only the steps for the hard goals, the algorithm repeatedly chooses some preference p, committing to adding the relevant actions to Π', and storing resulting length–cost pair. The choice of whether to include preference p' next, rather than p, is made at line 9, by comparing the relaxed plans obtained by including the relevant actions. There are several options, and we propose (and will evaluate) two:

Algorithm 2: Length–Cost Set Approximation

Data: Π: a relaxed plan
Result: LC: a set of length–cost pairs derived from Π
1 $\Pi' \leftarrow hard(\Pi)$;
2 $LC \leftarrow \{\langle |\Pi'|, cost(\Pi', \Pi) \rangle\}$;
3 $prefs \leftarrow P(soft(\Pi))$;
4 **while** $prefs \neq \emptyset$ **do**
5     $p \leftarrow$ first from $prefs$;
6     $best \leftarrow \Pi' \cup relevant(soft(\Pi), p)$;
7   **foreach** $p' \in (prefs \setminus \{p\})$ **do**
8         $new \leftarrow \Pi' \cup relevant(soft(\Pi), p')$;
9   **if** new *better than* $best$ **then**
10             $p \leftarrow p'$, $best \leftarrow new$;
11     $\Pi' \leftarrow best$;
12     $prefs \leftarrow omit(\Pi', \Pi)$;
13     $LC \leftarrow LC \cup \{\langle |\Pi'|, cost(\Pi', \Pi) \rangle\}$;
14 **return** LC

By length If $|new| < |best|$, then p' is better than p. This leads to the greedy selection of a preference that introduces the (joint) smallest number of new actions into the relaxed plan.

[1] P' is any member of the power-set $\mathcal{P}(P(soft(\Pi)))$

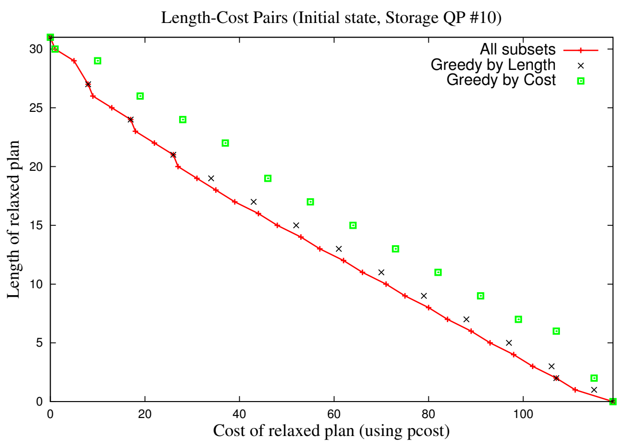

Figure 2: Length–Cost Pairs: All Subsets vs Greedy

By cost If $cost(new, \Pi) < cost(best, \Pi)$, then p' is better than p. This leads to the greedy selection of a preference that is (joint) best in terms of the cost of the resulting relaxed plan.

Note that in the case where two choices are tied according to the chosen metric, the other metric is used as a tie-breaker. For instance, if two preferences would lead to the equal-length relaxed plans, the preference that leads to the lower-cost relaxed plan is preferred. If the two options are tied according to this secondary metric, the tie is broken arbitrarily.

As an example of the length–cost pairs that can be obtained during search, Figure 2 shows the length–cost pairs for the initial state in problem 10 from the Storage Qualitative Preferences Domain, from IPC2006. The solid line is the Pareto front found by considering all subsets of $P(soft(\Pi))$, and the points denote using Algorithm 2 with the length and cost criteria. Both approaches follow the general trend of the Pareto front, with length doing particularly well in this problem: it is finding Pareto-optimal length–cost pairs towards the left- and right-edges of the graph.

4 Searching with Length–Cost Pairs

So far, we have discussed how length–cost pairs can be obtained from a relaxed plan, providing estimates of the number of actions needed to reach goal states of varying costs. In this section, we look at the issue of how to use these in anytime search, where we have an upper-bound on solution cost C, and are trying to find solutions better than this. In tasks with hard goals, initially $C = \infty$; in those without hard goals, C is the cost of preferences violated by the empty plan. As new solutions are found, C progressively decreases.

4.1 Managing Multiple Heuristic Estimates

Anytime search has two objectives: to try to reach the lowest cost goal states possible; and to reach goal states with cost better than C. To reach the best possible goal states, we have as a source of guidance the length of the relaxed plan Π from a state s to the goals, as produced by Algorithm 1. As Π includes actions to reach as many preferences as possible, $h_{all}(s) = |\Pi|$ is an estimate of the number of actions needed to obtain the best possible goal state reachable from s. To reach goal states better than C, as observed in previous work (Thayer et al. 2012; Thayer, Benton, and Helmert 2012), a promising approach is to prioritize states that are heuristically close to a goal state and appear to be low-cost enough to do better than C. In our work here, rather than having a heuristic estimate of the cost of reaching the goal and the length of the plan needed to do so, we have several paired heuristic estimates of these. From the length–cost pairs $LC(s)$ for a state s, reached by a plan of cost $g(s)$, the relevant relaxed plan length is:

$$h_{<C}(s) = \min(\{l \mid (l, c) \in LC(s) \wedge c + g(s) < C\} \cup \{\infty\})$$

The first of these two lines finds the smallest length which leads to a low-enough-cost goal state. The second covers the case where no such length exists, and hence the h value is taken to be infinity. As the length–cost pairs may be derived from non-admissible estimates of cost (e.g. the *nbcost* formula in Section 3.3), this does not necessarily mean the state should be pruned, just that $h_{<C}(s)$ is undefined.

As we now have multiple heuristic length estimates – h_{all} and $h_{<C}$ – the question is how to combine them. For this, we take our inspiration from the approach taken in Fast Downward (Helmert 2006), maintaining two open lists: one sorted by $h_{all}(s)$, the other by $h_{<C}(s)$. Search then alternates between the two, expanding the state at the head of the open list, and inserting the successors generated into both (removing them from both when expanded).

One crucial difference when using $h_{<C}(s)$ is that it is cost-bound dependent: when C changes, $h_{<C}(s)$ might change, also. This is reflected in Figure 2: each time the permissible cost of the relaxed plan is reduced by around 5, due to a tightening of C, the necessary relaxed plan length increases. To this end, each time C is reduced, we revisit the open list sorted by $h_{<C}(s)$, and check whether states' heuristic values have increased.

The algorithm to update this open list is shown in Algorithm 3. We assume a conventional bucketed open-list, with one bucket per h value, each containing a list of the states with that h value. Starting with the highest-index bucket, the algorithm checks whether the h values of any states in the bucket have changed, due to the new bound C (line 4). In the case where $h_{<C}(s)$ has increased, but is finite, s is removed from the open-list and re-inserted into the appropri-

Algorithm 3: Recalculating $h_{<C}(s)$ Values

Data: L: an open-list, a cost bound C
1 *upper* $\leftarrow$ highest-index bucket in L;
2 **foreach** $i \in [upper \ldots 0]$ **do**
3   **foreach** s *in* $L[i]$ **do**
4   **if** $h_{<C}(s) > i$ **then**
5   remove s from $L[i]$;
6   **if** $h_{<C}(s) \neq \infty$ **then** put s into $L[h_{<C}(s)]$;

	Soft Goals							Simple			Qualitative				Complex			
Planners	driver log	elev-ators*	open stacks*	path-ways	rov-ers*	peg-sol	SG Total	stor-sp	TPP sp	trucks sp	rovers qp	stor-ageqp	TPP qp	trucks qp	pathwa-ys-cnp	TPP cnp	SQC Total	Total
GBL	**11**	11	**30**	**27**	13	**30**	**122**	**10**	**14**	14	**12**	10	9	10	16	3	**98**	**220**
GBC	**11**	10	**30**	**27**	12	**30**	120	5	**14**	14	**12**	8	**12**	10	**21**	3	99	219
Baseline	9	10	8	16	10	**30**	83	**10**	13	11	6	**13**	11	9	9	**5**	87	170
All Subsets	**11**	12	9	19	14	**30**	95	6	4	**16**	5	5	5	**12**	9	**5**	67	162
HPlan-P	**11**	-	-	7	-	25	43	4	4	4	0	4	4	3	-	-	23	66
Lama2011-K&G	5	**30**	3	11	**17**	**30**	96	-	-	-	-	-	-	-	-	-	-	-
Lama2008-K&G	8	6	2	13	11	**30**	70	-	-	-	-	-	-	-	-	-	-	-
Baseline-K&G	6	4	2	4	5	**30**	51	-	-	-	-	-	-	-	-	-	-	-

Table 1: Comparison of New Planner Configurations to LAMA and the Baseline.
Figures show the number of problems on which each produced a (joint) best solution.

ate higher-index bucket. If $h_{<C}(s)$ is infinite, it is removed from this open list entirely – i.e. s will only then be expanded if it is chosen from the open list sorted by $h_{all}(s)$.

4.2 Selective Use of Length–Cost Pairs

Finding and using $h_{<C}$ values as described has a range of effects on search; but, likewise, has a number of side effects. The most obvious is the time and memory overheads: the additional computational effort of finding several length–cost estimates, and the extra space needed to store them with each state. To some extent, the overheads are lessened through the use of a greedy algorithm that considers and produces only a fraction of the possible length–cost pairs. Beyond this, though, there is the issue of what effect the alternation method described has on the states visited.

Taking as a starting approach searching with $h_{all}(s)$ alone, the states that are chosen for expansion are those that are heuristically closest to satisfying all hard goals and all preferences. Adding alternation to this, when states from the $h_{<C}$ open list are expanded, their successors are placed on the h_{all} open list (and vice versa). Thus, the states that would have been expanded before – if h_{all} alone had been used – will only now be expanded if their $h_{<C}$ or h_{all} values are not undercut by the newly considered states. In effect, the $h_{<C}$ open list introduces a systematic bias into the h_{all} open list, and into search in general.

Towards the start of search, such a bias is likely beneficial: searching by $h_{<C}$ has the effect of promoting states that are likely to lead to a reduction in C sooner than would be observed otherwise. The side-effects in terms of memory usage are also likely harmless: relatively few states and their associated heuristic values need to be stored in memory. Later in search, though, once C has tightened, $h_{<C}(s)$ approaches $h_{all}(s)$, as good solutions need to satisfy more of the preferences. Thus, the heuristic guidance is almost the same, and yet the computational and memory costs are higher.

To this end, we propose restarting search, reverting to searching according to $h_{all}(s)$, half-way through the time allowed for planning (empirically the results are fairly insensitive to the restart time chosen). As in previous work (Richter, Thayer, and Ruml 2010), the motivation is to eliminate the open-list bias; but also, in our case, such a configuration is not subject to the additional overheads of using $h_{<C}$.

5 Evaluation

We have implemented our techniques in the planner OPTIC (Benton, Coles, and Coles 2012), a state-of-the-art net-benefit planner that handles preferences natively. Since we are not reasoning with temporal problems in this work, the baseline configuration (standard OPTIC) can be thought of as a net-benefit implementation of LPRPG-P (which we do not use as it does not handle net-benefit domains), but with some overheads due to the temporal origins of the planner. It performs WA* search, with W=5, guided by h_{all}. We evaluated our planners on four different types of domains:

Soft-Goal Domains that can be solved by LAMA by using Keyder & Geffner's compilation of soft goals into action costs (2009). The domains used are those from the Net Benefit track of IPC2008; the Rovers 'Metric Simple Preferences' and 'Pathways Simple Preferences' domains from IPC2006; and a variant of the IPC2002 Driverlog domain, where each package has two conflicting goal locations, with random costs. We compare to LAMA 2008 and 2011 on these domains.

Simple Preference Domains (soft goals and precondition preferences) from IPC2006, for which compilations do not exist: a fully automatic implementation of Keyder and Geffner's approach is not available, and many of the domains use ADL, making manual translation impractical.

Qualitative Preference Domains (simple plus trajectory preferences) from IPC2006: these cannot be compiled, requiring a preference-aware planner. On these we compare to the baseline, and show in our table as a point of reference HPlan-P, the best performing domain-independent preference planner from IPC2006.[2]

Complex Preference Domains (allow numeric conditions) from IPC2006 – Pathways and TPP – with time removed.

All experiments were run on a 3.4GHz Core i7 machine running Linux, and were limited to 30 minutes of CPU time and 4GB of memory.

[2]We chose HPlan-P over SGPlan because circumventing the textual recognition of domains by SGPlan 6 (Hsu and Wah 2008) has been shown to significantly impact the performance of both SGPlan 5 and 6. When irrelevant textual changes are made HPlan-P performs better overall (Coles and Coles 2011).

	Soft Goal							Simple			Qualitative				Complex		Totals	
Planners	driver log	elev- ators*	open stacks*	path- ways	rov- ers*	peg- sol	SG Total	stor- sp	TPP sp	trucks sp	rovers qp	stor- ageqp	TPP qp	trucks qp	pathwa- ys-cnp	TPP cnp	SQC Total	All
GBL15-NB-B15	**15**	**25**	**26**	**30**	**18**	**30**	**144**	**13**	**17**	12	**15**	10	8	13	**25**	8	121	**265**
GBL15-NB-T-B15	**15**	24	19	**30**	15	**30**	133	10	**17**	**16**	13	**15**	**9**	**14**	23	**9**	**126**	259
GBL15-B15	**15**	15	19	**30**	16	**30**	125	**13**	**17**	12	**15**	10	8	13	**25**	8	121	246
GBL	**15**	17	19	**30**	16	**30**	127	9	10	12	12	7	**9**	10	1	1	71	198
Baseline	13	12	6	17	11	**30**	89	10	8	12	9	6	8	10	2	6	71	160
GBL-NA	**15**	12	11	11	15	28	92	5	6	3	9	5	5	2	2	3	40	132

Table 2: Comparison of Different Planner Configurations.
Figures show the number of problems on which each produced a (joint) best solution.

5.1 Length–Cost Set Approximation Algorithms

We first turn our attention to the algorithms for finding length–cost pairs, as described in Section 3.3. For simplicity, we start with a basic configuration of the planner: alternating between the h_{all} and $h_{<C}$ open lists, with costs from the $pcost$ cost function (Section 3.3), and without restarting after 15 minutes (Section 4.2). We compare three ways to find length–cost pairs: considering All-Subsets of preferences; choosing preferences Greedily by the Length metric (GBL); and choosing Greedily by Cost (GBC). We compare these to the Baseline, and where possible, both International Planning Competition versions of LAMA, with the Keyder & Geffner compilation. As we devote a later section of this evaluation to a comparison with LAMA, we focus first on comparison to the baseline.

The data obtained from our experiments are summarised in Table 1. The table shows the number of problems on which each of the planners found the best solution (including joint (equal) best solutions) in each domain. As expected the overheads of finding length–cost pairs by considering All Subsets of preferences is high. The exponential number of subsets means that much time is spent in the heuristic calculation, so the planner simply does not scale: it is unable to explore as many states within the time limit. The one exception to this is the two Trucks variants where the preferences have a negligible effect on relaxed plan length, even in larger problem instances, so enumerating the subsets of preferences is in fact feasible. In general, the two Greedy configurations are, however, the most successful: they have very similar overall performance, and out-perform both All Subsets and the Baseline. In some domains Greedy by Length is more successful, whilst in others Greedy by Cost dominates. The best planner for a given domain depends which strategy finds length–cost pairs closer to the Pareto front, as illustrated in Figure 2.

5.2 Search Configuration

Next we consider the effects of search configuration, rather than heuristic configuration, on the planner's performance. Due to the number of possibilities, a full cross-evaluation is not feasible. We therefore start with the best-so-far 'Greedy by Length' configuration, and note the effect of configuration changes on its performance. The relevant data are shown in Table 2. (Note that since the figures in the Tables are defined relative to the other planners listed in the table, data in the Tables 1 and 2 cannot be meaningfully compared.)

First, we look at **the effect of alternation** in search (described in Section 4.1). For this, we constructed a 'no alternation' configuration that repeatedly expands nodes on the $h_{<C}$ open list; and only once this is empty, considers those on the h_{all} open list. As can be seen in Table 2, 'GBL-NA' – greedy-by-length, no alternation – performs considerably worse than GBL – greedy-by-length (with alternation). Moreover, it is, overall, worse than the baseline configuration: it is better to search with h_{all} values alone. This again demonstrates the power of alternation: as in Fast Downward (Helmert 2006), combining two heuristics in this way offers far greater performance than either alone.

Next, we evaluate **the restart mechanism** proposed in Section 4.2. For this, we run Greedy-by-Length for 15 minutes; then the baseline planner for 15 minutes. This is shown in Table 2 as 'GBL15-B15'. The results are neatly divided according to the category of domain. In Soft Goal domains, the effect is marginal: the performance is slightly worse in Elevators. On the other domains, though, GBL15-B15 is markedly better: it finds the best (or equal best) solution 48 more times than GBL.

Making use of the inadmissible **net benefit cost** (NB) from the relaxed plan – the $nbcost$ cost defined in Section 3.3, rather than $pcost$ – leads to a further improvement in performance. Naturally this is only apparent in the domains with action costs (marked * in Table 2), as otherwise, $nbcost \equiv pcost$. We take the current best configuration (GBL15-B15) and add net-benefit (GBL15-NB-B15). The gains seen in the domains with action costs are impressive, demonstrating that the planner can indeed benefit from this inadmissible but informative measure.

The final configuration we tested, given we have multiple heuristic values, was **tie-break according to** h_{all}, when inserting states into the $h_{<C}$ open list. In line with previous work using a *preferredness* mechanism to break ties between equal-h-valued states (Richter and Helmert 2009), this tie-breaking favours states that are preferable according to h_{all}. We take the current best configuration (GBL15-NB-B15) and add tie-breaking to it (GBL15-NB-T-B15), presenting the results in Table 2. Unfortunately, the results here are inconsistent: while there are gains in a number of Simple Preference and Qualitative Preference domains, there are significant losses on Soft Goal domains with action costs. With the exception of Rovers Qualitative Preferences this

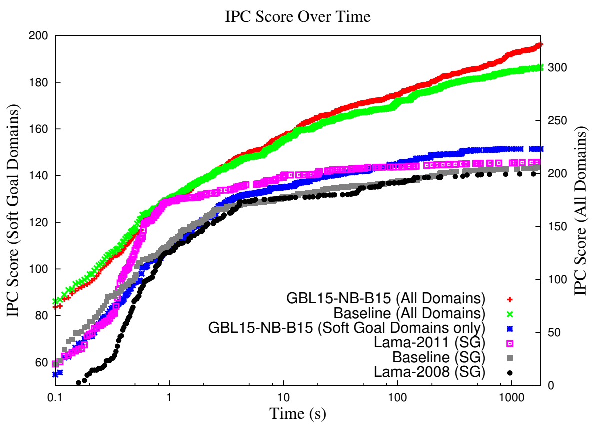

Figure 3: IPC score vs Time. The order of the legend corresponds to the order in which the lines meet the y-axis.

suggests that heuristic tie-breaking is effective in domains without action costs, but not in those with action costs. This leaves an interesting question for future work: determining whether an effective tie-breaking scheme can be found.

5.3 Comparison to Lama on Net-Benefit Domains

As noted in previous work, soft goals can be compiled away (Keyder and Geffner 2009), and the resulting compilation used with efficient, cost-minimising planners. In 2009, this compilation, used with the then-latest version of Lama, represented the state-of-the-art in planning with soft goals. Lama 2011, the winner of the last International Planning Competition, is equally suitable, and offers better performance still. Contained within Table 1 is an interesting result, found during our experiments: the Baseline planner – which represents solely previous work – outperforms Lama 2011 with compiled-soft-goals in all but two domains.

Looking at where the differences lie, between Lama and the baseline, Lama does well on domains with a strong net-benefit component coming from action costs. The domains with action costs are marked * in Table 1. When choosing which states to expand during search, the baseline planner pays little attention to action costs: it is guided by h_{all}, which estimates the number of actions needed to satisfy all the preferences, regardless of action cost. The two domains in which Lama produces better solutions than the Baseline (or incidentally, the other configurations in both tables) are the domains in which action costs are more important in determining plan quality. Conversely, in the other domains, the best quality plans are found by achieving the right combination of soft goals. The third domain with action costs – Openstacks – is more suited the other planners as overall solution cost is dominated by making the correct choice of soft goals to achieve: action costs contribute only a small proportion of the cost. The overall picture is that the domains in which the planner has to make interesting trade-offs between soft goals to achieve the best cost, rather than simply achieving as many as possible then minimising costs, are those in which the other planners tested outperform Lama the most.

To determine how our new approach compares to Lama we directly compared our current-best configuration (GBL15-NB-B15) to Lama 2011, on the Soft Goal domains. In this comparison, GBL15-NB-B15 found the best solution 56 times, and Lama found the best solution 21 times. On another 71 problems, the solution costs were equal – mostly on smaller problem instances, and in Pegsol where optimal solutions are found by both approaches. This result shows that native reasoning about soft-goals is a worthwhile endeavour and can lead to performance improvements. (We note that the result still holds if we exclude our soft-goal variant of Driverlog, and use only pre-existing domains.)

As a further comparison, we took the domain and problem files used by Lama (Keyder & Geffner compilations), and evaluated them using the Baseline planner. The resulting performance is shown as 'Baseline-K&G' in Table 1. Excluding Pegsol (where it is comparatively easy to find optimal solutions), it performs consistently worse than the other planners. This confirms that the action-cost-optimisation prowess of Lama is considerably better than our approach, and that our strengths lie in reasoning with preferences explicitly. It also alludes to the relative implementation efficiencies of the two planners: Lama 2011 is not a temporal-planner-derivative, with the overheads that entails.

5.4 Anytime Behaviour

The final aspect of our algorithms that we consider is their anytime behaviour: how the quality of solutions progresses over time. To normalise for the different magnitudes of solution costs in different domains, we use the quality metric introduced in IPC2008. The score for planner p on task i at time t is:

$$score(p, i, t) = best\_cost(i) \div cost(p, i, t)$$

$best\_cost(i)$ is the cost of the best solution found to i (by any planner in $<$ 30mins), and $cost(p, i, t)$ is the cost of the solution found by planner p by time t. To reward quality and coverage, if p has not solved i by time t, $score(p, i, t) = 0$. The cumulative IPC score by a planner p at time t is then:

$$score(p, t) = \sum_{i \in benchmark\_tasks} score(p, i, t)$$

Figure 3 shows the anytime progression of IPC scores of four planners: GBL15-NB-B15, Baseline, and Lama 2011 and 2008. To allow meaningful comparisons to be made, two sets of results are shown. The bottom four lines correspond to running these planners only on the Soft Goal domains, accessible to Lama. The top two lines correspond to running GBL15-NB-B15 and Baseline on all domains. (Note the different y-axis scales for each of these).

It is interesting to note that initially (up to about 2 seconds) the Baseline configuration performs very slightly better than GBL15-NB-B15. This reflects the computational costs of producing the initial heuristic values: in many tasks, at first, applying just a few plan steps yields a better solution, so the extra heuristic guidance does not pay off. This effect though is soon reversed: after around 2 seconds, GBL15-NB-B15 is always the better option. When using all domains, there is a noticeable jump in score at around 900 seconds, where search restarts, with the final score reached being 7% higher than the baseline. Consistent with Table 2, restarting offers no benefits on Soft Goal domains. One final observation is that although Lama 2011 outperforms the Baseline configuration in terms of number of joint best solutions found (Table 1), it performs very similarly in terms of IPC score after 30 minutes.

6 Conclusion

In this paper, we considered the challenge of guiding search through the goal-dense search spaces encountered in planning tasks with preferences. We presented a new cost-bound-sensitive heuristic that promotes states that appear to be close to a new-best goal state. Used in alternation with an existing distance-to-go measure based on achieving all reachable preferences, our heuristic leads to substantial improvement in the quality of solutions found. In future work, we will explore the use of our approach in combination with cost-to-go heuristic measures, to improve performance in domains with action costs.

References

Baier, J.; Bacchus, F.; and McIlraith, S. 2007. A heuristic search approach to planning with temporally extended preferences. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

Benton, J.; Coles, A. J.; and Coles, A. I. 2012. Temporal planning with preferences and time-dependent continuous costs. In *Proceedings of the Twenty Second International Conference on Automated Planning and Scheduling (ICAPS)*.

Benton, J.; Do, M. B.; and Kambhampati, S. 2009. Anytime heuristic search for partial satisfaction planning. *Artificial Intelligence* 173:562–592.

Coles, A. J., and Coles, A. I. 2011. LPRPG-P: Relaxed Plan Heuristics for Planning with Preferences. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*.

Coles, A. J.; Coles, A. I.; Clark, A.; and Gilmore, S. T. 2011. Cost-sensitive concurrent planning under duration uncertainty for service level agreements. In *Proceedings of the 21st International Conference on Automated Planning and Scheduling (ICAPS)*.

Cushing, W.; Benton, J.; and Kambhampati, S. 2011. Cost-based satisficing search considered harmful. In *Proceedings of the Third ICAPS Workshop on Heuristics for Domain-independent Planning*.

Do, M. B., and Kambhampati, S. 2003. Sapa: Multi-objective Heuristic Metric Temporal Planner. *Journal of Artificial Intelligence Research* 20:155–194.

Edelkamp, S.; Jabbar, S.; and Nazih, M. 2006. Large-Scale Optimal PDDL3 Planning with MIPS-XXL. In *IPC Booklet, ICAPS*.

Fox, M., and Long, D. 2003. PDDL2.1: An Extension of PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research* 20:61–124.

Gerevini, A. E.; Long, D.; Haslum, P.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic Planning in the Fifth International Planning Competition: PDDL3 and Experimental Evaluation of the Planners. *Artificial Intelligence* 173:619–668.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research* 26:191–246.

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

Hsu, C., and Wah, B. 2008. SGPlan 6 source code, http://ipc.informatik.uni-freiburg.de/planners, file seq-sat-sgplan6/parser/inst_utils.c, lines 347–568. Accessed September 2010.

Keyder, E., and Geffner, H. 2009. Soft goals can be compiled away. *Journal of Artificial Intelligence Research* 36:547–556.

Pohl, I. 1973. The avoidance of (relative) catastrophe, heuristic competence, genuine dynamic weighting and computation issues in heuristic problem solving. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Richter, S., and Helmert, M. 2009. Preferred Operators and Deferred Evaluation in Satisficing Planning. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS)*.

Richter, S., and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *Journal of Artificial Intelligence Research* 39:127–177.

Richter, S.; Thayer, J. T.; and Ruml, W. 2010. The Joy of Forgetting: Faster Anytime Search via Restarting. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling (ICAPS)*.

Richter, S.; Westphal, M.; and Helmert, M. 2011. LAMA 2008 and 2011. In *IPC Booklet, ICAPS*.

Smith, D. E. 2004. Choosing objectives in over-subscription planning. In *Proceedings of the 14th International Conference on Automated Planning & Scheduling (ICAPS)*.

Thayer, J. T., and Ruml, W. 2011. Suboptimal Search: A Direct Approach Using Inadmissible Estimates. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

Thayer, J.; Benton, J.; and Helmert, M. 2012. Better Parameter-free Anytime Search by Minimizing Time Between Solutions. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*.

Thayer, J. T.; Stern, R.; Felner, A.; and Ruml, W. 2012. Faster Bounded-Cost Search Using Inadmissible Estimates. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling (ICAPS)*.