Using the Relaxed Plan Heuristic to Select Goals in Oversubscription Planning Problems

Angel García-Olaya, Tomás de la Rosa, and Daniel Borrajo *

Universidad Carlos III de Madrid,
Avda de la Universidad 30, 28911, Leganés, Spain
www.plg.inf.uc3m.es

Abstract. Oversubscription planning (OSP) appears in many real problems where finding a plan achieving all goals is infeasible. The objective is to find a feasible plan reaching a goal subset while maximizing some measure of utility. In this paper, we present a new technique to select goals “a priori” for problems in which a cost bound prevents all the goals from being achieved. It uses estimations of *distances* between goals, which are computed using relaxed plans. Using these distances, a search in the space of subsets of goals is performed, yielding a new set of goals to plan for. A revised planning problem can be created and solved, taking into account only the selected goals. We present experiments in six different domains with good results.

1 Introduction

In classical planning the objective is to find a sequence of actions transforming a given initial state into a final state in which a conjunctive list of goals is present. A valid plan is the sequence of actions reaching all goals. Soft goals can be added to a classical planning problem to account for goals that we wish to achieve but that we do not enforce. As a result, a planning problem could contain both hard and soft goals. In real domains there can be several causes making impossible or useless to reach all soft goals: two or more soft goals could be mutually exclusive, i.e. cannot be true at the same time; plans achieving all goals could need more quantity of a certain resource than the available one; goals could be redundant, so a plan would be valid even achieving just some of them; or some goals could be not worth enough, as the cost of achieving them would be higher than their reward. In problems with only soft-goals, a valid plan is any plan achieving any subset of them, even an empty one. Usually, an utility or penalization is assigned to each soft goal to compare plans achieving different sets of soft goals.

Oversubscription planning (OSP) is a special type of planning with soft goals, introduced by [13], and motivated by some real problems at NASA. OSP assumes it is not possible to achieve all soft goals due to a resource limitation; the rover

* This work has been partially supported by MICIIN TIN2008-06701-C03-03 and CCG10-UC3M/TIC-5597 projects.

battery power in the original formulation. A simple way to model the limited resource is as the maximum cost a valid plan can have. The objective is to find a plan that maximizes the utility while keeping the cost (resource) under a certain bound. We will call it the COST-BOUNDED OSP problem to distinguish it from the more general OSP case, where there can be other causes preventing all the goals from being achieved (mutually exclusive goals for example). The objective of OSP COST-BOUNDED problems is to find the plan with maximum *utility* given the resource(s) availabilities. The *utility* is a function, generally the addition, of the utilities of the goals reached by a plan.

A close related soft goals problem is the Partial Satisfaction one (PSP) [12, 1, 2] included in the International Planning Competition (IPC) 2006 under the PREFERENCES Track and in the IPC 2008 under the NET-BENEFIT track. In PSP, nothing prevents, at least a priori, achieving all goals, but there is a trade-off between the utility of achieving a goal and the cost of doing so [1]. The most explored PSP problem is the NET-BENEFIT, which tries to maximize the $utility - cost$ metric (actually it assigns a penalization to each not reached goal and minimizes the $cost + penalization$). Probably due to the IPC, work on NET-BENEFIT has been extensive. In contrast, the COST-BOUNDED problem has been less explored, even though the existence of a limited resource (time, fuel, battery, storage space, money...) preventing the accomplishment of all goals is present in a large number of real domains.

The optimal solution for a soft goals problem can be computed by finding a plan for each of the 2^n combinations of the n problem's goals, and then selecting the plan with maximum utility. Of course, this is infeasible except for very simple cases. In practice, three different approaches have been used: a priori selection of goals, on-the-fly selection, and compilation into a different problem. Goals can be selected "a priori" to find the potentially best subset to plan for. Later, the planning step takes into account only the selected goals. In [13], an orienteering problem (OP) is constructed. A set of propositions, different for each domain, make up the nodes of the basic OP. For each goal a node is added, inserting an arc from it to the nodes where it can be achieved. Arcs costs are calculated using a plan graph. The resulting OP is solved using beam search and the solution is used to guide a partial-order planner. This is the only approach tailored for resource-bounded problems found in the literature. The main disadvantage is that the set of propositions making up a node depends on a threshold that has to be manually defined for each domain. In [12], relaxed plans are used to estimate the cost of reaching a goal, but for the NET-BENEFIT problem. For a $n - goals$ problem, a relaxed plan to each goal is computed to estimate the NET-BENEFIT of including it in the set of goals. Then, for each goal, a set containing it is constructed adding goals until the NET-BENEFIT does not increase. On-the-fly selection of goals approaches do not perform goals selection. Instead, incremental plans are built, refining the best ones to achieve more utility [4, 1]. In general, these approaches scale worse than the goal selection ones. A soft goals problem can be modeled as a Markov Decision Process (MDP) [13, 2], obtaining a policy from which a plan finding the optimal solution can be extracted. However, this conversion does not scale well so it has not been reported to be used in practice. Using integer programming (IP) to find optimal plans for a given parallel plan length is another possible transformation [2]. The most successful problem transformation [9] compiles away soft goals to create a STRIPS+actions cost problem with more actions, fluents and hard goals, but no soft goals, which can be solved by any conventional planner. This compilation, in combination with the winner of the satisficing track of the IPC 2008 [11], outperforms any participant of the soft-goals tracks of the two last IPCs [9].

The technique we propose is a two-step algorithm. In the first step we select the goals to plan for by using relaxed plans to compute distances among goals. Distances represent estimated costs of achieving one goal from a state where another one has been achieved. Using those distances, we perform a search in the space of subsets of goals for the set that maximizes the utility with the estimated cost-bound. In the second step, this set is given to a satisficing planner to find a plan achieving it.

In the following section we will formally define the problem. Next, the two-step algorithm will be described, and our technique will be compared with previous work. The paper finishes with conclusions and future work.

2 Problem definition

We will define next the planning problem we are tackling.

Definition 1: *A* STRIPS *planning problem with actions costs and soft goals is a tuple $P = \{F, A, I, G, c, u\}$, where F is a finite set of fluents, A is a finite set of actions, being each $a_i \in A$ composed of preconditions establishing when the action can be applied, and effects, consisting of elements of F being added or deleted from the current state after a_i is applied, $I \subseteq F$ is the initial state, $G \subseteq F$ is the set of goals, $c : A \mapsto \mathbb{R}_0^+$ is a cost function, and $u : G \mapsto \mathbb{R}^+$ is an utility function.*

A solution of the planning problem P is an ordered list of actions $\Pi = \{a_0, a_1...a_n\}, a_i \in A$, which applied to I results in a state where $G' \subseteq G$ is true (in classical planning, $G' = G$). If the final state is forced to achieve some $G'' \subset G'$ then we have a problem with both hard and soft goals. The cost of the plan Π is defined as $C(\Pi) = \sum_{a_i \in \Pi} c(a_i)$. The objective of a planning problem with soft goals is usually to maximize the utility of the plan. We will consider additive utilities, as most work in the field (see [4] for other approximations).

Definition 2: *The utility of a solution, Π, to the planning problem with soft goals is $U(\Pi) = \sum_{g_i \in G'} u(g_i)$.*

Definition 3: *A* COST-BOUNDED *problem is a tuple $M = \{P, \mathcal{C}_{max}\}$, where P is a planning problem with soft goals as defined above, and $\mathcal{C}_{max} \in \mathbb{R}^+$ is the cost bound of the problem.*

A solution to the COST-BOUNDED problem is a plan $\Pi = \{a_0, a_1...a_n\}, a_i \in A$, which applied to I results in a state where $G' \subseteq G$ is true, and such that its *plan cost* satisfies $C(\Pi) \leq \mathcal{C}_{max}$. Given two solutions for the COST-BOUNDED problem Π_1 and Π_2, Π_1 will be a better solution than Π_2 if $U(\Pi_1) > U(\Pi_2)$.

3 Two-step OSP Algorithm

We perform an *a priori* selection of goals in two steps: selecting goals and planning for the selected goals. Algorithm 1 shows the pseudo-code of the process.

Algorithm 1 Two-step algorithm for solving COST-BOUNDED problems.

```
OSP (P OSP problem ): $\Pi$ Plan
    $S \leftarrow$ Select-goals($P$)
    $P' \leftarrow$ standard problem (no OSP) from new goals($P, S$)
    $\Pi \leftarrow$ Plan-for-goals($P'$)
return $\Pi$

Select-goals (P OSP problem ): $S$ Goals set
    $D \leftarrow$ Compute distances matrix
    $S \leftarrow$ Select goals($D$)
return $S$

Plan-for-goals (P planning problem ): $\Pi$ plan
repeat
    $\Pi \leftarrow$ plan($P$)
    if $\Pi \neq \emptyset$ then
        return $\Pi$
    else
        $P \leftarrow$ remove lowest utility goal from $P$
    end if
until goals($P$) $= \emptyset$
return Fail
```

3.1 Selecting Goals

In the first step, we generate a matrix of distances between goals. This matrix has $n + 1$ rows and n columns, being n the number of goals. The elements of the first row are the estimations of the cost of reaching each goal from I, as if each goal were the only goal in the problem. The following rows, one for each goal, contain the estimations of the cost of achieving the remaining goals from the state reached when calculating the first row.

Definition 4 (Distance from the initial state to a goal): *Let P be a planning problem and $g_x \in G$ a goal. We define the distance from I to g_x (Δ_{Ix}) as the cost of the lowest cost plan, Π^*_x reaching g_x from I.*

The value of Δ_{Ix} gives an idea about how close a given goal and the initial state are, i.e. how costly to reach a single goal from the initial state is. Therefore, it makes it easier to decide whether to include or not this goal in the set of goals to plan for. In most cases, achieving one goal will change the cost of reaching others, which is accounted for by means of the following distance between two goals:

Definition 5 (Distance between two goals): *Let P be a planning problem, $g_x \in G$ a goal, Π^*_x the lowest cost plan used to compute Δ_{Ix}, $s_{\Pi^*_x}$ the state resulting from applying Π^*_x to I, and $g_y \in G$ another goal. The distance from g_x to g_y (Δ_{xy}) is defined as the cost of the lowest cost plan reaching g_y from $s_{\Pi^*_x}$.*

So, in order to compute the distance between two goals, the lowest cost plan computed in the previous step is applied to the initial state to reach another state where the first goal is achieved, and then a distance to the second goal is computed in the same way as before. In general, $\Delta_{xy} \neq \Delta_{yx}$.

Both Δ_{Ix} and Δ_{xy} depend on the calculation of the lowest cost plan with only one goal. Even if all the other goals are removed, computing this plan is usually difficult, making the computation of these distances infeasible and discouraging their use. As an example, we tried to use an optimal planner to compute Δ_{Ix} for the propositional Rovers domain. It was only possible to compute it in the first 22 (out of 40) IPC5 problems.

Instead, an approximation of Δ_{Ix} (Δ'_{Ix}) can be computed using relaxed plans. For the rest of the paper, we will compute Δ'_{Ix} as the cost of the non-optimal relaxed plan reaching g_x from I, in a similar way as Metric-FF does [8]. In order to compute Δ'_{xy} the relaxed plan extracted to compute Δ'_{Ix} is applied to I to reach a state in which g_x is true. Obviously, quite often, actions belonging to the relaxed plan will not be applicable, as some of their preconditions will not be satisfied. Despite this fact, actions are applied ignoring preconditions and only taking into account the effects. From this state, the distance to g_y is computed, using again the cost of the non-optimal relaxed plan. A mutex check should be done before calculating Δ'_{xy}, but as there are no mutex goals in the domains we have tried, we have not implemented it yet.

Once the distances matrix is generated, we use a beam search algorithm to find the goal set with higher utility in the space of subsets of goals. The root node is the empty set. In the first step, the k goals with higher utility are selected, being k the beam width. For each of the selected goals g_x, we annotate the node with the corresponding Δ'_{Ix} and $u(g_x)$. In the second step we consider all the combinations of the previously selected k goals and one of the remaining goals and annotate the accumulated cost and utility for these two-goals sets. If a set is composed of g_1 and g_2, the utility will be $u(g_1)+u(g_2)$ and the cost $\Delta'_{Ig_1}+\Delta'_{g_1g_2}$. We select the k best sets and so on. Search ends when a set including all the goals has been found, or, more likely, when it is not possible to add goals to any of the k best sets without exceeding the cost-bound. In this case, the set with higher utility is returned as the solution of the search process. We break ties favoring lower estimated cost. In case of further tie, one is picked arbitrarily. This algorithm is greedy in the sense that once a goal is selected for inclusion in a planning set, it is always considered in the same relative order with respect to the other goals in that set.

Regarding search parameters, we have tried with different beam widths (0.25, 0.5, 1, 5, 10, 50, 100 and 500 times the number of goals) and 5 gives the best results in most domains, although variations in utilities depending on beam width do not seem to be very high.

3.2 Planning for the Selected Goals

Once goals have been selected, we generate a new problem with the goals in the selected goals set. The new problem is given to a Metric-FF-like planner, CBP [6]. Its performance is comparable to LAMA for many domains and, unlike it, it allows numeric preconditions, which are present in all COST-BOUNDED domains.

Given that *distances* (i.e. costs) are estimated, often the list given to the planner is still oversubscribed. To solve this problem, we try to find a plan during a given time bound. If no plan is found, the lowest utility goal is removed and we search for a new plan. This is repeated until a valid plan has been found or all the goals have been removed. To find a plan, the planner is given the same time used to calculate distances, with a minimum of 10 seconds, which has shown experimentally to work well.

3.3 Computational Complexity

For a *n-goals* problem, we have to extract n non optimal relaxed plans to compute $\varDelta'_{Ix}$, which is polynomial in time [7]. Then we apply these relaxed plans to obtain the new initial states for each goal, which is linear in the number of steps of the relaxed plans. Next, for each goal g_x, we have to create *n-1* relaxed plans to compute $\varDelta'_{xy}$. This gives us $n + n \times (n-1)$ relaxed plans, so it is quadratic in the number of relaxed plans. Comparing with [12], in the worst case, when low oversubscription, the complexity of their approach in terms of relaxed plans and goals is $n \times \sum_{i=1}^{n-1} i \approx n^3$ relaxed plans. Furthermore, we are always solving relaxed plans with only one goal while they incrementally construct relaxed plans with two, three, …goals. If only a few of the goals can be achieved, the complexity of their algorithm decreases dramatically, while ours remains constant. In addition, we have to select the goals. Nodes at the first layer of the search graph include only one goal. In the second, they have two goals, and so on. Thus, the maximum depth will be $n-1$ in case all the goals except one can be achieved. Given that we have chosen a beam search width of $5n$, the complexity is quadratic in n. In comparison, complexity of [13] OP is exponential on the number of propositions defining a node, which depends on a manually selected threshold and varies in each domain.

4 Experimental Results

We have tested our technique in six IPC domains. To create COST-BOUNDED problems, first we have taken the PREFERENCES or NET-BENEFIT versions, removed the preferences part to make them regular actions cost domains and solved them using a Metric-FF like satisficing planner. The aim is to have an upper bound for the plan cost. In domains where no such versions exist we have used the STRIPS + actions cost version, solving it in the same way. Second, equivalent problems with a cost bound of 25%, 50% and 75% of the previously computed cost have been generated. These three values allow to study how well our technique performs when there is a high (25%), medium (50%) or low (75%) oversubscription degree. Domains have been changed by adding a new fluent to account for the cost bound. For any action increasing the cost of the plan a new precondition has been added. This precondition prevents the action from being applied if its cost, plus the current accumulated cost, exceeds the cost bound. For example, if the cost of a problem solution is 100, three new problems have been created. These problems have the same initial state and goals than the original one, but their maximum cost is limited to 25, 50 and 75 respectively. We did not use the penalizations of the original problems because we were interested in testing whether different distributions of utilities among goals yield different results. Instead, we have defined two versions of each problem; in the first one all the goals have the same utility: $u(g_i) = 1, \forall g_i \in G$. In the second one, the utility of each goal is a random value between 1 and 10: $1 \leq u(g_i) \leq 10, \forall g_i \in G$.

We have compared our approach, that we will call *Distances*, with the most similar current work: an adaptation for COST-BOUNDED problems of Keyder et al. compilation [9]; Mips-XXL [5], ranked second in the NET-BENEFIT track of the IPC 2008 (the winner exhausts the memory even with the simpler problems); SGPlan [3], winner of the PREFERENCES track of the IPC 2006; and a *Baseline* planner which greedily selects the goal with higher utility and plans for it. If a plan is found, the two goals with higher utility are selected and so on. Compiled, Mips-XXL and SGPlan are tailored for the NET-BENEFIT problem and not for the COST-BOUNDED one. That means that in the search process they will try to minimize the penalization for not achieving the soft goals, but in general the heuristic will go *blind* with respect to the cost bound. A way to tackle this is to modify the metric, so the problem is converted into a kind of NET-BENEFIT-COST-BOUNDED one, i.e. both the total cost and the penalizations have to be minimized. But a focus has to be put on the penalizations as against the cost; the planner should not avoid reaching a goal even if its utility is lower than its cost given that the cost is not higher than the cost bound. As a preliminary version, we have changed the metric of the compiled problems from *(minimize (+ (penalizations-cost)* to *(minimize (+ (penalizations-cost)(/ (plan-cost) (cost-bound))*, which slightly improves their performance

Domains tested are *Rovers* from IPC5, and *Driverlog* and *Depots* from IPC3. *Rovers* is a good example of a domain where goals can not be undone once achieved and there are not strong interactions among goals. *Depots* has been chosen because goals can be undone and there are many interactions among them. In *Driverlog*, in addition, a significant percentage of the goals are present at the initial state and the planner must undo them to find a valid plan. We have also tested *Transport*, *Peg Solitaire* and *Elevators* domains from the IPC 2008 NET-BENEFIT track. *Crewplanning* has universal quantifiers not supported by our planner. *Openstacks* is mainly an optimization domain in which the cost of a good plan is very low, making it difficult to create different degrees of oversubscription. And *Woodworking's* soft goals are not a single predicate but a conjunction of them, which is not supported yet by our approximation. For the experiments we have used a Intel Xeon 3Ghz with 3GB of RAM memory and a time bound of 900 seconds. For the *Compiled* problems, the planner uses the whole 900 seconds to find and refine the plan. The same applies for Mips-XXL, which, although being an optimal planner, is able to generate intermediate non-optimal plans. In contrast, for *Baseline*, SGPlan and the *Distances* version, no plan refining is done; the first found plan is returned.

Table 1 shows the results. Scores for each planner are calculated in a similar manner as in the IPC: the planner finding the plan with higher utility (U_{max}) gets 1 point. Every other planner scores U/U_{max}. SGPLan has been removed from the table as it only solves problems in two domains (*Peg Solitaire* and *Rovers*) and even in these domains the quality is quite poor. The best result for each domain and oversubscription degree is highlighted in bold.

	Baseline			*Distances*			*Compiled*			*Mips-XXL*		
Domain	25%	50%	75%	25%	50%	75%	25%	50%	75%	25%	50%	75%
$Depots_1$ (22)	12.9	14.0	14.8	**16.6**	**16.0**	**16.4**	14.8	14.2	15.8	14.9	12.5	11.4
$Depots_{10}$ (22)	12.7	14.8	14.5	**15.9**	**17.6**	**16.7**	13.1	13.1	14.9	14.9	13.0	12.7
$Driverlog_1$ (20)	11.1	12.0	13.3	**16.4**	**17.8**	**18.1**	15.9	15.4	14.8	13.5	14.1	12.3
$Driverlog_{10}$ (20)	11.8	12.6	14.3	**17.0**	**18.0**	**18.9**	16.1	15.7	14.7	14.0	14.7	13.1
$Elevators_1$ (30)	8.0	22.0	22.6	23.5	25.8	28.1	**26.0**	**29.2**	**28.2**	1.5	0.8	0.8
$Elevators_{10}$ (30)	11.4	19.9	26.5	22.8	25.4	27.4	**26.0**	**28.7**	**27.6**	23.7	19.2	13.5
$Pegsol_1$ (30)	20.2	19.9	20.4	24.9	27.8	29.0	**28.2**	**29.6**	**29.8**	**28.2**	28.8	27.0
$Pegsol_{10}$ (30)	21.0	21.2	22.2	27.3	28.8	29.1	**29.0**	**29.7**	**29.9**	**29.0**	28.6	27.6
$Transport_1$ (30)	11.0	14.8	19.9	**15.5**	**18.8**	**21.7**	13.0	17.3	18.1	0.0	0.0	0.0
$Transport_1$ (30)	7.7	16.5	23.4	**15.0**	**21.4**	**24.6**	12.6	18.4	19.9	0.0	0.0	0.0
$Rovers_1$ (20)	10.5	15.9	17.6	15.2	16.9	**19.2**	**20.0**	**19.7**	19.0	16.3	11.6	9.0
$Rovers_{10}$ (20)	10.6	15.9	18.3	14.1	16.9	**19.2**	**20.0**	**19.7**	18.3	17.8	13.8	11.3
Total	148.8	199.4	227.7	222.7	250.8	268.6	234.6	250.8	250.9	173.6	157.0	138.6

Table 1. Results on quality. Number next to each domain is the number of problems, i.e. the maximum score a planner can get. High (25%), medium (50%) and low (75%) oversubscription rates have been considered.

Baseline performs always worse than *Distances*, except in some domains, especially with low oversubscription, where it performs closer. In general, IPC soft goals domains tend to have a low number of goals, most of the times less than ten. In this case, the greedy approach of *Baseline* performs close to other approaches when the cost bound is high. In domains that have problems with a higher number of goals, like *Driverlog* or *PegSolitaire*, the differences are much bigger. We plan to create more complicated problems to see if this tendency continues. *Distances* performs better than *Compiled* in 20 problem sets and worse in 16, while Mips-XXL is better only in low oversubscription *PegSolitaire.* Different utility profiles make no significant difference, but degree of oversubscription does. *Distances* performs better in low oversubscription domains in 8 out of 12 configurations. In high and medium oversubscription degrees there is a tie; both approaches behave better in 6 out of 12. In these problems, the low *cost-bound* prunes quite quickly the search tree, allowing a more complete exploration by the iteratively refining algorithm. As soon as the maximum cost increases, yielding a bigger search space, selection of goals by *Distances* returns better results. Again, we expect that in more complicated problems these differences will be magnified and *Distances* will outperform *Compiled.*

Time cannot be easily compared as *Compiled* and Mips-XXL use the whole 900 seconds to refine the solutions, while the other planners finish as soon as a valid plan has been found. Table 2 shows the accumulated time needed to find the best solution for *util = 1*, problems (results for $1 \leq util \leq 10$ problems are similar). For *Compiled* it means time spent to find the last solution within the 900 seconds limit (so, not necessarily consuming all the time). *Mips-XXL* is not included in the comparison as there is no way to know when the partial solutions are generated. *Baseline* is usually the fastest one, though in some domains *Distances* is better. *Compiled* is most of the times the slowest one.

	Baseline			*Distances*			*Compiled*		
Domain	25%	50%	75%	25%	50%	75%	25%	50%	75%
Depots	189	255	265	1391	1826	1788	4306	3851	5051
Driverlog	139	203	214	585	879	799	436	1951	3593
Elevators	47	253	321	19	122	185	22	3005	4638
PegSolitaire	10	80	122	910	1525	1766	224	3266	3401
Rovers	189	207	209	59	145	86	141	2617	2779
Transport	131	212	274	195	345	354	1145	3513	2655
Total	706	1209	1405	3160	4842	4978	6273	18202	22118

Table 2. Accumulated total time in seconds to find the best solution.

5 Conclusions and Future Work

In this paper we have presented a method to solve COST-BOUNDED oversubscription problems based on the computation of a *distance* between goals using relaxed plans. This distance indicates how far two goals are, allowing to search in the space of subsets of goals to find a subset maximizing utility with an estimated cost lower than a given cost bound. To find plans for this subset we have used a planner with performance comparable to the winner of the last IPC.

We have evaluated this approach against NET-BENEFIT planners as no other COST-BOUNDED planner is, to our knowledge, freely available. Problems with high, medium and low oversubscription have been created by limiting the cost a plan can have to 25%, 50% and 75% of the estimated total cost. Results show that our technique offers better quality in problems with low oversubscription and in domains where the medium number of goals is above ten. In problems with high or medium oversubscription or with low number of goals, its performance is comparable with the best technique; Keyder's et al. compilation. Our technique is also almost always much faster than the compilation.

In the future, we plan to implement smarter strategies to apply when the selected goals set is still oversubscribed, or when the real cost of the found plan is lower than the maximum cost, allowing thus for more goals to be achieved. In our current implementation the planner does not take any advantage of the order in which the goals were selected. We want to explore whether biasing the planner to follow this order would increase the performance. Some ways to do that are, for example, to modify the heuristic values of nodes, or to use a goal agenda as presented in [10].

We plan also to make experiments in other domains and in more complicated problems, as those of the sequential satisficing track of the IPC. In the current configuration, for NET-BENEFIT planners (*Compiled*, *SGPlan* and *Mips-XXL*), the effect of the metric is to take into account the cost of the plan as another goal (for util=1 problems) or as the lowest utility goal (for $1 \leq$ util ≤ 10 problems). We want to experiment with different metrics to guide the NET-BENEFIT planners in a different way.

References

1. Benton, J., Do, M., Kambhampati, S.: Anytime heuristic search for partial satisfaction planning. Artificial Intelligence 173, 562–592 (2009)
2. van den Briel, M., Sanchez, R., Do, M.B., Kambhampati, S.: Effective approaches for partial satisfaction (over-subscription) planning. In: Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-2004). pp. 562–569 (July 2004)
3. Chen, Y.X., Wah, B.W., Hsu, C.W.: Temporal planning using subgoal partitioning and resolution in sgplan. J. of Artificial Intelligence Research 26, 323–369 (2006)
4. Do, M.B., Benton, J., van den Briel, M., Kambhampati, S.: Planning with goal utility dependencies. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07). pp. 1872–1878. Hyderabad, India (2007)
5. Edelkamp, S., Jabbar, S.: Mips-xxl: Featuring external shortest path search for sequential optimal plans and external branch-and-bound for optimal net benet. In: Proc. 2008 International Planning Competition. Sydney, Australia (2008)
6. Fuentetaja, R., Borrajo, D., Linares, C.: A look-ahead B & B search for cost-based planning. In: Proceedings of the Thirteenth Conference of the Spanish Association for Artificial Intelligence. pp. 105–114 (2009)
7. Hoffman, J., Nebel, B.: The ff planning system: Fast plan generation through heuristic search. Journal of Artificial Intelligence Research 14, 253–302 (2001)
8. Hoffmann, J.: The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. Journal of Artificial Intelligence Research 20, 291–341 (2003)
9. Keyder, E., Geffner, H.: Soft goals can be compiled away. Journal of Artificial Intelligence Research 36, 547–556 (2009)
10. Koehler, J., Hoffmann, J.: On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. Journal of Artificial Intelligence Research 12, 338–386 (2000)
11. Ritcher, S., Westphal, M.: The lama planner: Guiding cost-based anytime planning with landmarks. Journal of Artificial Intelligence Research 39, 127–177 (2010)
12. Sanchez-Nigenda, R., Kambhampati, S.: Planning graph heuristics for selecting objectives in over-subscription planning problems. In: Proceedings of the 15th Intl. Conf. on Automated Planning & Scheduling (ICAPS-05). pp. 192–201 (2005)

13. Smith, D.: Choosing objectives in over-subscription planning. In: Proceedings of the 14th Intl. Conf. on Automated Planning & Scheduling. pp. 393–401 (2004)