Over-Subscription Planning with Numeric Goals

J. Benton
Computer Sci. & Eng. Dept.
Arizona State University
Tempe, AZ 85287-5406
j.benton@asu.edu

Minh B. Do
Embedded Reasoning Area
Palo Alto Research Center
Palo Alto, CA 94304
minh.do@parc.com

Subbarao Kambhampati[*]
Computer Sci. & Eng. Dept.
Arizona State University
Tempe, AZ 85287-5406
rao@asu.edu

Abstract

By relaxing the hard-goal constraints from classical planning and associating them with reward values, over-subscription planning allows users to concentrate on presenting what they want and leaves the task of deciding the best goals to achieve to the planner. In this paper, we extend the over-subscription planning problem and its limited goal specification to allow numeric goals with continuous utility values and goals with mixed hard and soft constraints. Together they considerably extend the modeling power of goal specification and allow the user to express goal constraints that were not possible before. To handle these new goal constraints, we extend the *Sapa^{ps}* planner's planning graph based techniques to help it choose the best beneficial subset of goals that can include both hard or soft logical and numeric goals. We also provide empirical results in several benchmark domains to demonstrate that our technique helps return quality plans.

1 Introduction

In classical planning, a given set of conjunctive goals must all be achieved together for the plan to be a success. As planning problems get more complicated with temporal and resource constraints, it is harder to predict which goals are achievable. Recently, motivated by the requirements of different NASA planning problems such as planning for airborne, telescopes such as Hubble and SIRTF, and planning for a Mars Rover mission, Smith [2003; 2004] introduced the over-subscription planning problem. In this type of problem, the planner is not required to achieve all the goals but instead achieve the *best* subset of goals given the resource limitations.

While the ability to represent goals as soft constraints with (fixed) utility values makes it much easier to model goals in many applications, the restriction to only support soft logical goals has limitations in metric temporal planning domains. In this paper, we extend the framework to handle numeric goals having variable utility values and mixed goal sets containing both hard and soft goal constraints. We first present a motivating example before discussing our techniques to handle the new extensions.

Mars Rover: A Rover is sent to Mars to collect soil/rock samples and take pictures at different locations. The rover should first land safely, roll off the lander and then go to different locations to gather scientific data. While the objective is to collect as much data as possible given energy and time limitations, only a subset of the data can be gathered. The quality of the plan will be measured by the amount of samples collected at different locations and the amount of picture data transferred back to earth. Obviously, more samples and picture data give better values and thus it is more natural to model the goals of collecting them as numeric goals with the utility given according to the amount actually collected.

The example above brings up two issues: (1) there are certain types of goals that are better represented as numeric goals with a range of utility values (e.g. amount of sample/data); (2) there are goals that are critical to the success of the plans (e.g. landing safely) and there are goals that are not critical but beneficial (e.g. collecting samples). Besides the rover domain, those types of goals are also common in many other real-world planning problems such as transportation or travel planning. We extend the goal structure by allowing numeric goals (e.g. constraints on some continuous quantities) with continuous utility values that depend on the actual value achieved by the final plan. We also strike a middle ground between classical planning and over-subscription goal constraints by supporting both hard (critical) and soft (beneficial) goals. Those additional types of goal constraints complement traditional logical goals and allow the user more freedom in expressing what they desire from the final plan.

While extending the current framework of hard and soft goals to support a mixture of them is not overly challenging, effective handling of the numeric goals with continuous utility values does pose several challenges. In particular, we need to reason about the *degree of satisfaction* of goals. To illustrate, if the goal is $Sample \geq 5\ grams$, it can be satisfied by $Sample = 6$ as well as $Sample = 10$ at different degrees. We need techniques to:

1. Assign utilities to different degrees of satisfaction.

2. Track costs of achieving goals of different degrees of satisfaction.

3. Use the achievement costs and utilities in combination

to estimate the final plan that maximizes the overall net benefit achievable from a given state.

We present $Sapa^{\mathcal{M}ps}$ an extension of $Sapa^{ps}$ [Do & Kambhampati, 2004; van den Briel *et al.*, 2004] to support both numeric goals and mixed soft/hard goal sets. $Sapa^{\mathcal{M}ps}$ significantly extends the relaxed plan extraction procedure used in $Sapa^{ps}$ to support numeric goals.

We first briefly discuss the search framework used in $Sapa^{ps}$ in Section 2. We then show how to extend the cost propagation and relaxed plan extraction routines to handle metric quantities and the combination of soft and hard goals in Section 3. We present empirical results on extended versions of some well-known planning benchmark domains in Section 4 to show that the new techniques help find larger and better quality plans. We conclude the paper with the related work and discussion.

2 Background

We start with the formal definition of the over-subscription (aka partial-satisfaction) planning problem. We then proceed by describing the $Sapa^{ps}$ [Do & Kambhampati, 2004; van den Briel *et al.*, 2004] planner and its framework that heuristically searches for good quality plans for the problem.

2.1 Over-Subscription Planning (OSP)

To formally define the over-subscription planning problems [Smith, 2004; van den Briel *et al.*, 2004], the following notations will be used: $\mathcal{F}$ is a finite set of fluents and $\mathcal{A}$ is a finite set of actions, where each action consists of a list of preconditions and a list of add and delete effects. $\mathcal{I} \subseteq \mathcal{F}$ is the set of fluents describing the initial state and $\mathcal{G} \subseteq \mathcal{F}$ is the set of goals. Hence we define a planning problem as a tuple $\mathbf{P} = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$. The OSP problem is defined as follows:

Definition OSP NET BENEFIT: Given a planning problem $\mathbf{P} = (\mathcal{F}, \mathcal{A}, \mathcal{I}, \mathcal{G})$ and, for each action $a \in \mathcal{A}$ a "cost" $c_a \geq 0$ and, for each goal specification $g \in \mathcal{G}$ a "utility" $u_g \geq 0$: find a finite sequence of actions $\Delta = \langle a_1, ..., a_n \rangle$ starting from $\mathcal{I}$ that leads to a state[1] $\mathcal{S}$ maximizing the net benefit value $\sum_{g \in \mathcal{S}_\mathcal{G}} u_g - \sum_{a \in \Delta} c_a$, where $\mathcal{S}_\mathcal{G}$ is the set of goals satisfied in $\mathcal{S}$.

Thus, in contrast to the classical planning problems, in OSP no goal achievement is needed to qualify the plan as valid (any sequence of actions is a valid plan). We thus put emphasis on finding *good quality* plans where the objective function is to maximize the tradeoff between total achieved goal utility and total incurred action cost (both additive).

2.2 $Sapa^{ps}$: Heuristic search for OSP

$Sapa^{ps}$ adapts the heuristic (progression) search framework to solve OSP problems. The quality of each search node S visited by the A* search algorithm in a forward planner is ranked using the value $f(S) = g(S) + h(S)$ with: $g(S) = \sum_{g \in \mathcal{S}_\mathcal{G}} u_g - \sum_{a \in \Delta} c_a$. Starting from the initial state $\mathcal{I}$ with the net benefit value $g(\mathcal{I})$, $Sapa^{ps}$ keeps track of all the visited states S that have better $g(S)$ values than the best

---

[1]States are represented just as in the Sapa planner [Do & Kambhampati, 2003].
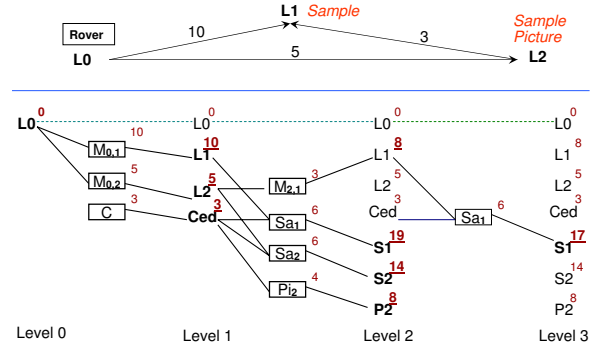

Figure 1: Rover example and the planning graph

state visited before S. Thus, instead of finding a single plan, the algorithm keeps searching for incrementally better plans in terms of the achieved net benefit ($g(S)$) value. The heuristic value $h(S)$ is calculated by extending the cost propagation over the planning graph routine. It is followed by the relaxed plan extraction process in *Sapa* [Do & Kambhampati, 2003]. The search stops when the first node in the queue has value $h(S) = 0$ (i.e. $f(S) = g(S)$). For the rest of this section, we will discuss the three steps in estimating the $h(S)$ value.

Example: Figure 1 shows an example in the Mars Rover domain along with the complete planning graph structure for this example[2]. The rover initially resides at location L_0 and needs to collect samples at L_1, and to take a picture of L_2. The rover can: (1) move between two locations (M_{L_x, L_y}); (2) calibrate the equipment before taking sample/picture (C); (3) collect a sample at location l (Sa_l); and (4) take pictures (Pi_l). Action execution costs are depicted next to each action in the complete planning graph for the example shown in Figure 1. The first action layer of the graph contains three actions with their respective execution costs: $C_{M_{0,1}} = 10$, $C_{M_{0,2}} = 5$, and $C_C = 3$.

Cost-propagation over the planning graph: The planning graph propagates the achievement cost of each predicate and execution cost of each action forward starting at the initial level/time-point until fix-point. To simplify the discussion for this section, we will assume that all actions have unit duration.

Intuitively, the heuristic needs to realize that to be able to take the picture at L_2 we first need to have the camera calibrated and be at L_2. For each action a, its execution cost C_a is static and different from the cost to enable its execution $cost(a)$, which is the cost to achieve all of its preconditions. Thus, $cost(Pi_{L_2}) = cost(Ced) + cost(L_2) = C_C + C_{M_{0,2}} = 5 + 3$. We also want the heuristic to capture the fact that the cost to have a picture involves the cost to enable execution of Pi_{L_2} and the cost to actually carry out that action. Thus: $cost(P_{L_2}) = cost(Pi_{L_2}) + C_{Pi_{L_2}} = 8 + 4$. The propagation rules are:

1. Initially, at $t = 0$: (1) $\forall p \in \mathcal{I} : cost(p) = 0$; (2) $\forall p \notin \mathcal{I} : cost(p) = \infty$; (3) $\forall a \in \mathcal{A} : cost(a) = \infty$.

2. At level l: (1) $\forall a \in \mathcal{A} : cost(a_l) = \sum_{p \in Precond(a)} cost(p_{l-1})$; (2) $\forall p \in \mathcal{F} : cost(p_l) =$

---

[2]We assume that the readers are familiar with the planning graph, which was originally described in [Blum & Furst, 1997].
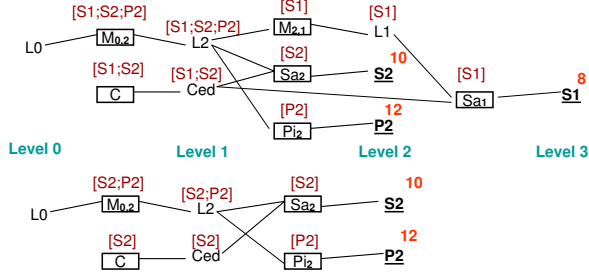
Figure 2: The relaxed plan

$$min_{p \in Effect(a)} cost(a_{l-1})$$

Our heuristic is inadmissible due to the use of sum propagation in calculating the cost to execute each action. As we grow the graph, new actions and facts are introduced and the cost to achieve facts ($cost(p)$) and execute actions ($cost(a)$) decreases due to new ways to achieve and support them. In Figure 1, we highlight the new facts and actions introduced at each level and the new achievement costs for each fact. For example, L_1 first appears at level 1 with $cost(L_1) = 10$ (achievable by $M_{0,1}$). The value of $cost(L_1)$ decreases to 8 at level 2 due to the new action $M_{2,1}$ in level 1. While the set of achieved facts and supporting actions in level 2 and level 3 are the same, we did not stop growing the graph at level 2. This is because $cost(S_1)$ decreases at level 3 due to the reduction in $cost(L_1)$ at level 2 (which leads to the decrease in $cost(Sa_1)$).

Extracting the relaxed plan: After terminating the cost-propagation, the cost values can be used to extract the relaxed plan RP, starting from all achieved goals G, as follows:

1. Starting from the last level, $\forall g \in G$ at level l select action a at the action level $l-1$ that supports g with the lowest cost.

2. When action a is selected at level l, then all preconditions p of a at the previous level will be added to G.

3. Stop when: $G \subset \mathcal{I}$.

All the collected actions a and the causal links between them make up the relaxed plan RP.

Refining the relaxed plan: For each goal g, we build the goal supporting set GS for each proposition p and action a by going backward using the extracted relaxed plan as follows:

- $\forall g \in G : GS(g) = \{g\}$
- $GS(a) = \bigcup GS(p) : p \in Effect(a)$
- $GS(p) = \bigcup GS(a) : p \in Precond(a)$

Intuitively, for each action a, $GS(a)$ is the set of goals that a supports. Thus, the achievement of any goal in $GS(a)$ depends on the inclusion of a in the relaxed plan while for any goal $g \notin GS(a)$, g will still be achievable without a. In Figure 2, we show the goal supporting sets for all actions and related propositions (e.g. $GS(C) = \{S_1, S_2\}$) and the corresponding utility values (e.g. $Util(S_2) = 10$) of the three goals in the relaxed plan. For each set $S_G \subset G$, let $A_{S_G} = a : GS(a) \subset S_G$ be the set of actions supporting only goals in S_G. We will remove S_G along with A_{S_G} from the relaxed plan if $\Sigma_{a \in A_{S_G}} cost(a) > \Sigma_{g \in S_G} Util(g)$ (i.e. cost > utility).

In our ongoing example, only the goal set $S = \{S_1\}$ can be removed because $cost(A_{S_1}) = cost(M_{2,1}) + cost(Sa_1) = 3 + 6 > Util(S_1) = 8$. Figure 2 shows the relaxed plan before and after refinement.

The net benefit of the final relaxed plan is: $Util(\{S_2, P_2\}) - cost(\{M_{0,2}, C, Sa_2, Pi_2\}) = (10 + 12) - (5 + 3 + 6 + 4) = 4$. This is used as the heuristic value $h(S)$ to guide the A* search algorithm. In general, we define $RP(S)$ to be the relaxed plan found for state S and $U(Apply(RP(S), S))$ to be the utility achieved by applying all actions in the relaxed plan to state S. The heuristic value is then calculated as: $h(S) = (U(Apply(RP(S), S)) - U(S)) - cost(RP(S))$ where $RP(S)$ is the final relaxed plan.

3 Handling Numeric Goals with Utility

$Sapa^{ps}$ only supports logical achievement goals of the form $g = True$ (e.g. $HasPicture(L_2)$). However, if we have the goal to collect at least m grams of a Mars soil sample at a given location l (i.e. $sample(l) > m$), we can more naturally represent it as a numeric goal.

3.1 Numeric Goal Representation

Unlike logical goals that only have true/false values, there are an arbitrarily large number of values that can satisfy a given numeric goal. We assign a range of *continuous utility values* for numeric goals to represent a *degree of satisfaction*. Specifically, numeric goals and their utility values are set up as follows:

Definition NUMERIC GOAL: A numeric goal is a relation $f \in D$ in which f is a mathematical formula involving an arbitrary number of numeric variables and $D = [l, u]$ is an interval open or closed at either end and is bounded by two real values $l \leq u$ (l, u can be infinity).

For example, the goal of keeping travel cost between \$100 and \$500 can be represented as: $Hotel + AirTicket \in [100, 500]$.

For each numeric goal $g \in D$, the utility value $u(g)$ is specified by a linear function. For example, if the goal is to collect at least 10 grams of Mars sample (i.e. $f = Sample > 10$), then the utility of this goal can be $u(f) = 10^6 * Sample$ (i.e. it is worth 1 million dollars for each gram of Mars' soil if we have at least 10 grams, but 0 dollars otherwise).

3.2 Cost Propagation with Numeric Goals

To incorporate numeric goals into our current heuristic framework, we first have to be able to estimate the cost to achieve them. Unlike logical goals, there are multiple degrees of satisfaction for a numeric goal g. Therefore, the procedure that tracks the achievement cost for g is necessarily more complicated. Specifically:

- For logical values we track the time point t_p at which a proposition p can first be achieved ($p = True$) and the achievement cost for p at time points $t \geq t_p$. For numeric values we need to track a *range* of possible values $[\mathsf{L}_j^{v_i}, \mathsf{U}_k^{v_i}]$ for a numeric variable v_i at each jth update to the lower bound value and k^{th} update to the upper bound value.
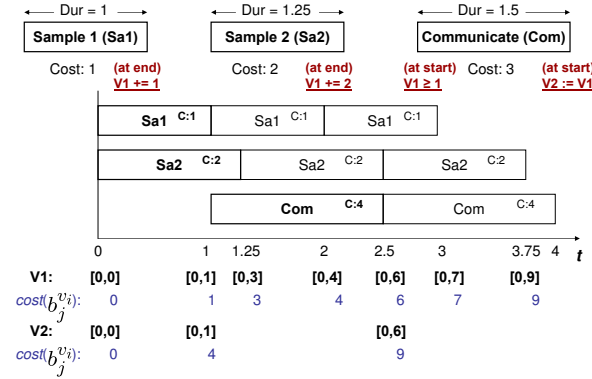
Figure 3: The RTPG for our example. Our actions are defined above it.

- In tracking costs to achieve logical literals, actions are only *re-applied* (e.g. Sa_1 at action level 2 in Figure 1) when the cost to execute (i.e. cost to achieve their pre-conditions) them decreases. However, actions having numeric effects on v_i need to be applied whenever possible (e.g. Sa_1 in Figure 3) because their effects continue to change the upper ($\mathsf{U}_k^{v_i}$) or lower ($\mathsf{L}_j^{v_i}$) bound values of the quantity v_i.

Example: To illustrate the techniques to track achievement costs for numeric goals, we will use a variation of our ongoing Mars Rover example. We solely concentrate on metric quantities in this example (illustrated in Figure 3). There are two sample-collecting actions: *sample1* (Sa_1) collects a single gram of soil sample; *sample2* (Sa_2) collects 2 grams of soil sample. The effects of these actions occur at the end of execution. The third action, *communicate* (Com), communicates the sample information to a lander at the start of execution. We use two continuous variables; v_1 to track the weight of the collected soil sample in the Rover's store and v_2 to track the total amount of communicated sample. The goal g is to achieve $v_2 > 5$ and the goal utility formula is $u(g) = v_2 * 3$ (i.e. if $v_2 \le 5$ we get a utility of zero, otherwise the utility is found using $u(g)$).

While the connection between time and numeric goals is not obvious in OSP problems, one important component of action cost is the amount of time consumed by each action. Goal utilities also normally depend on the time the goals are achieved. Like $Sapa^{ps}$, the $Sapa^{\mathcal{M}ps}$ planner handles actions with different durations and thus we do not make the assumption that all actions have uniform duration (as in previous section).

Tracking the upper and lower bounds: The first step in estimating the achievement costs for numeric goals is to track the possible values for numeric variables at different time points. Previous work in tracking upper/lower bounds for numeric variables using the planning graph was done for non-temporal planning [Koehler, 1998; Hoffman, 2003; Sanchez & Mali, 2003]. There, actions are either always executed serially or marked mutually exclusive of one another if their numeric effects give varying results when ordered differently. In $Sapa^{\mathcal{M}ps}$, the semantics of the planner disallows interacting actions to be concurrent, whereas the relaxed temporal planning graph (RTPG) allows this. In our context, this

means that an action in the RTPG can be applied concurrently with itself an arbitrary number of times, causing the number of time points in the graph to increase significantly. To avoid this problem, we disallow such concurrency as a practical compromise.

Figure 3 shows a RTPG for our example. We re-apply the numeric effects of actions directly after their duration completes. At time $t = 1$, action Sa_1 completes and we add the upper bound value $\mathsf{U}_1^{v_1} = 1$ of v_1 to indicate that the collected weight of the soil sample has increased by 1. Also at this time point, the precondition of Com can be satisfied by the bound $\mathsf{U}_1^{v_1} = 1$, so we apply the numeric effects of the action, adding the upper bound $\mathsf{U}_1^{v_2} = 1$ on the *weight* of the soil sample. At $t = 1.25$, Sa_2 completes and we add the upper bound value $\mathsf{U}_2^{v_1} = 1 + 2 = 3$ increasing the previous upper bound of v_1 the second time by 2 according to the numeric effect of Sa_2. This continues until we reach our numeric goal $v_2 > 5$ (when ($\mathsf{U}_2^{v_2} = 6$) > 5). In Figure 3, we show the upper and lower bound values for v_1 and v_2 as we grow the graph. Because we do not have actions that decrease the values of v_1 or v_2 in this simple example, the lower bound values of those two variables remain unchanged.

The RTPG handles numeric expressions in effects and preconditions by applying the formulas to each bound. For instance, if the bounds of three variables v_1, v_2, v_3 are $v_1 : [0, 20]$, $v_2 : [1, 5]$ and $v_3 : [-1, 3]$ and we want to calculate $f = v_3 + v_1 * v_2$, we first find $v_1 * v_2 = [min(0 * 1, 0 * 5, 20 * 1, 20 * 5), max(0 * 1, 0 * 5, 20 * 1, 20 * 5)] = [0, 100]$ then $f = v_3 + [0, 100] = [0 + -1, 100 + 3] = [-1, 103]$.

Tracking achievement costs: We let $b_j^{v_i}$ refer the jth upper or lower bound of v_i in the RTPG. The RTPG associates for each bound value[3], $b_j^{v_i}$, a propagated cost, $cost(b_j^{v_i})$. The cost value estimates how costly it is to achieve a certain numeric value. The idea is that for each value n that satisfies some numeric goal, the tradeoff between the cost of achieving n and the utility that n incurs will be used as heuristic guidance.

Cost propagation is not trivial in the presence of numeric expressions. Before turning to this case, let us concentrate our discussion on simple numeric effects (i.e. effects using only constant values). Numeric updates (e.g. increases and decreases) will generate a new bound value with respect to the previous one. Because of this, we base the cost of each bound on the previous bound's cost, $cost(b_{j-1}^{v_i})$. Specifically, when an action a adds an upper or lower bound for a variable v_i using an increase (+=) or decrease (-=) numeric effect, the propagated cost of the bound is $cost(b_j^{v_i}) = cost(a) + cost(b_{j-1}^{v_i})$. This lets us track the cost of executing several actions that may be required to reach a numeric goal or precondition. Bounds found using an assign (:=) numeric effect only depend upon the action itself. So, the propagated cost is $cost(b_j^{v_i}) = cost(a)$. In our example shown in Figure 3, the cost of the new bound found by the numeric effect of Sa_2 at $t = 1.25$ is $cost(\mathsf{U}_2^{v_1}) = cost(Sa_2) + cost(\mathsf{U}_1^{v_1}) = 3$. When the numeric effect of Sa_1 is re-applied at $t = 2$ and causes

---

[3]The RTPG can find multiple upper and lower bound values for a variable v_i at the same time point. We index the bounds in this manner so that we may refer to previously found bounds without regard to the time point that they may appear in the RTPG.

the third update on the value of v_1 we have $cost(\mathsf{U}_3^{v_1}) = cost(Sa_1) + cost(\mathsf{U}_2^{v_1}) = 4$.

In our ongoing example, each numeric effect and precondition involves only two or fewer variables. However, in a more general scenario, numeric goals, action preconditions and action effects can be a formula involving an arbitrary number of numeric variables. These inter-dependencies between variables further complicates cost propagation on bounds. That is, when we calculate new bound values from expressions, we need to find the cost of the new bound based upon the costs of the values involved in the expression. To do this, we define for each variable v_i, a set $\mathcal{B}_{v_i}$ of all bound values involved in computing a new bound for variable v_i. For example, we have an expression $f = v_3 + v_1 * v_2$ and define an effect $v_4 += f$. When applying this effect, we track each of the bounds used to generate the minimum and maximum values from this formula (in this case [-1, 103]). We apply the effect using the resulting bounds. For the new upper bound of v_4, we have the set $\mathcal{B}_{v_4} = \{\mathsf{U}^{v_1} = 20, \mathsf{U}^{v_2} = 5, \mathsf{U}^{v_3} = 3\}$ to indicate the dependencies between the upper bound of v_4 and the particular bound values of v_1, v_2, and v_3 used to achieve this new upper bound.

During cost propagation, bounds used to satisfy a precondition are included in the cost of an action. For each variable used in expressions, we allow only a single bound of that variable to be included when calculating costs of new bounds (i.e. avoid including more than one bound for each variable when a variable is used both to support a precondition and as part of an expression in a numeric effect). To do this, we let $\mathcal{P}_a$ be the set of all bound values used to satisfy the numeric preconditions of an action a. In our example, $\mathcal{P}_{Com} = \{\mathsf{U}^{v_1} = 1\}$ starting at $t = 1$. We also define $\mathcal{P}_a \overline{\bigcap} \mathcal{B}_{v_i}$ to be a set operation over the variables represented by the bounds in $\mathcal{P}_a$ and $\mathcal{B}_{v_i}$, where the result gives us only the bounds in $\mathcal{P}_a$ that are not equal to the bounds in $\mathcal{B}_{v_i}$. More formally, $\mathcal{A} \overline{\bigcap} \mathcal{B} = \{b_j^{v_i} : b_j^{v_i} \in \mathcal{A} \wedge \exists b_{k \neq j}^{v_i} \in \mathcal{B}\}$.

For instance, if we have the sets $\mathcal{P}_a = \{\mathsf{L}_1^{v_1} = 1, \mathsf{U}_1^{v_2} = 2\}$ and $\mathcal{B}_{v_i} = \{\mathsf{U}_3^{v_1} = 6, \mathsf{U}_1^{v_2} = 2, \mathsf{U}_1^{v_3} = 1\}$ then $\mathcal{P}_a \overline{\bigcap} \mathcal{B}_{v_i} = \{\mathsf{L}_1^{v_1} = 1\}$. This result provides the means for removing the costs of bounds in $\mathcal{P}_a$ that are already present in an action, when we also use them in $\mathcal{B}_{v_i}$ to calculate bound cost. Thus, the cost of a bound value of variable v_i that is changed by an increase (+=) or decrease (-=) effect of an action a is $cost(b_j^{v_i}) = cost(a) + cost(b_{j-1}^{v_i}) + \sum_{d \in \mathcal{B}_{v_i} \setminus \mathcal{P}_a} cost(d) - \sum_{e \in \mathcal{P}_a \overline{\bigcap} \mathcal{B}_{v_i}} cost(e)$. It follows that the cost of a bound found by an assign effect (:=) on action a is $cost(b_j^{v_i}) = cost(a) + \sum_{d \in \mathcal{B}_{v_i} \setminus \mathcal{P}_a} cost(d) - \sum_{e \in \mathcal{P}_a \overline{\bigcap} \mathcal{B}_{v_i}} cost(e)$.

As shown in Figure 3, at $t = 0$ both actions Sa_1 and Sa_2 are added to the RTPG. The delayed numeric effect of Sa_1 increases v_1 by 1 at $t = 1$ and incurs $cost(\mathsf{U}_1^{v_1} = 1) = 1$. At this time point, the precondition $v_1 \geq 1$ of Com is satisfiable and we have $\mathcal{P}_{Com} = \{\mathsf{U}_1^{v_1} = 1\}$. The $cost(\mathsf{U}_1^{v_1} = 1)$ is included to find $cost(Com)$. So, when Com is put in the graph at $t = 1$, we have $cost(Com) = 4$. Its instantaneous effect $v_2 := v_1$ leads to a new bound for v_2 with $cost(\mathsf{U}_1^{v_2} = 1) = cost(Com) = 4$. At $t = 1.25$, Sa_2's delayed numeric effect is activated and increases v_1 by 2 to a new bound with cost $cost(\mathsf{U}_2^{v_1} = 3) = 3$. As we increase the time to $t = 2$

and $t = 2.5$, we keep updating the upper bound of v_1 and v_2 to values as shown in Figure 3. When activating Com at $t = 2.5$, we have the set $\mathcal{B}_{v_i} = \{\mathsf{U}_4^{v_1} = 6\}$ representing the bounds used for the expression $v_2 := v_1$. We get a new bound that satisfies the numeric goal with $cost(\mathsf{U}_2^{v_2} = 6) = cost(Com) + cost(\mathsf{U}_4^{v_1} = 6) - cost(\mathsf{U}_1^{v_1} = 1) = 9$. At $t = 3$ the update of Sa_1 completes, giving $cost(\mathsf{U}_5^{v_1} = 7) = 7$ and at $t = 3.75$, the numeric effect of Sa_2 gives $cost(\mathsf{U}_6^{v_1} = 9) = 9$.

Notice that even after numeric goals $f \in [L, U]$ are satisfied by the bound values on f at a given time point t (i.e. $[L, U] \bigcap [l^f, u^f] \neq \emptyset$ at t), we allow the RTPG to continue to expand until fix-point.

Relaxed Plan Extraction with Numeric Goals: After doing cost-propagation over the RTPG, the cost information can be used to extract a relaxed plan using an approach similar to that discussed in Section 2.2. The challenge here is in deciding for each numeric goal $f \in [l, u]$, how to select the most beneficial value v_f of f that is achievable through the planning graph and extract the action that supports that v_f value. When selecting an action a we add all of its logical and numeric preconditions into the goal set G. Also, we ensure that the cost of numeric bounds used to satisfy the numeric goal constraints are included with the relaxed plan. This is so we can accurately determine the achievement cost for each bound.

To handle relaxed plan extraction for numeric goals, we choose the bound values that provide the best tradeoff between goal utility value[4] with the achievement cost. Thus, for each achievable value $v_g \in [\mathsf{L}_j^{v_g}, \mathsf{U}_k^{v_g}]$ that satisfies the goal constraint on g, we select the one that gives the greatest $U(v_g) - cost(v_g)$ value. The action a that supports v_g is then selected and added to the relaxed plan. If given a goal interval $v_i \in [l, u]$ we never find a value $v \in [l, u]$ while expanding the RTPG, but do find two values $v_l < l$ and $v_u > u$, we say the goal is subsumed. In this case, we allow the first subsuming value to support the goal. However, since we cannot estimate the utility on this bound, we let its utility be 0.

Heuristic estimation: For each numeric goal supporting bound that we select, we include the cost to support it and its utility value in the net benefit calculation. In our ongoing example the net benefit of the relaxed plan would be $U(\{u(v_2 = 6)\}) - (C(Sa_2) + C(Com) + C(\mathsf{U}_4^{v_1} = 6) - C(\mathsf{U}_1^{v_1} = 1)) = (6 * 3) - (2 + 4 + 6 - 1) = 7$.

3.3 Combining Hard & Soft Goals

In our work, we support hard and soft goals for both traditional logical goals and the numeric goals discussed in the previous section. In the case of numeric goals, a single goal may involve both hard and soft constraints. For example the goal of having the Rover collect between 5 to 10 grams of Mars soil ($SC = SoilCollected \in [5, 10]$), can be modeled so that $SC > 5$ is a hard constraint (i.e. should collect at least 5 grams) and $SC < 10$ is a soft constraint (i.e. 10 is enough but more than that is not harmful).

To support both hard and soft goals in the best first search framework for over-subscription problem, we need to change

---

[4]Recall that the utility values are given by a linear formula over the goal variable, g.

both the search-termination criterion and the heuristic estimation routine in the search framework discussed in Section 2.2. Specifically:

Search node evaluation: When all goals are soft, any visited node encodes a legal plan. In the presence of hard goals, only nodes that satisfy all hard goals can be represented a valid plan.

Heuristic estimation:

- For a given node, if we cannot achieve some hard goal when building the planning graph until fix-point for cost propagation, then the node is a deadend.

- When refining the relaxed plan, we only remove soft goals and actions (solely) supporting them.

Notice that with the mixed soft/hard goal combination, it's possible to return plans that have negative net benefit (total action cost is higher than the total achieved goal utility). Those plans with negative net benefit will not be found if all the goals are soft constraints because an empty plan with zero net benefit is a valid plan in that case.

4 Empirical Evaluation

We have implemented the support for numeric goals on top of the $Sapa^{ps}$ planner and also extended its best-first-search framework to support the combination of hard and soft goals. We call the new planner $Sapa^{\mathcal{M}ps}$. To test the implementation, we have created a test suite by extending the problem sets of the *Mars Rover* and *Logistic* domains. Both were used in previous planning competitions (IPC2002 and IPC2000). The experiments aim at testing whether or not the cost-propagation techniques for numeric goals can help $Sapa^{\mathcal{M}ps}$ find reasonable size plans with good quality for the extended OSP problems.

4.1 Configuration of Testing Domains

All test suites involve durative actions consuming metric resources. Action costs are added to the action representation and appear as a function of the time and resources consumed by that action (e.g. $cost(travel) = HotelRate * Duration(travel) + AirTicket$). Utilities of goals are randomly generated within some reasonable bounds. Logical goals have fixed utilities while utilities of numeric goals are represented using linear functions. The goals have the same probability to be either *hard* or *soft* (applied to both logical and numeric goals). Specific types of goals in the domains are created with various probabilities. In general the potential number of numeric goals increases with the problem number, with a minimum of at least one numeric goal.

The *Rovers* domain is extended and includes the weight of collected soil and rock samples. The goals are to send analyzed data about samples at different locations back to the lander. Utility of a sample is proportional to the weight collected for the sample. Rover's store has a fixed weight capacity and each time a sample action is executed, a single gram is stored. If the Rover keeps collecting sample, then the cost of sampling outweighs the overall utility of the collected sample at some certain time.

In the *Logistics* domain, we add numeric goals to deliver a certain number of packages of the same type to its destination using trucks and airplanes. The initial state is extended
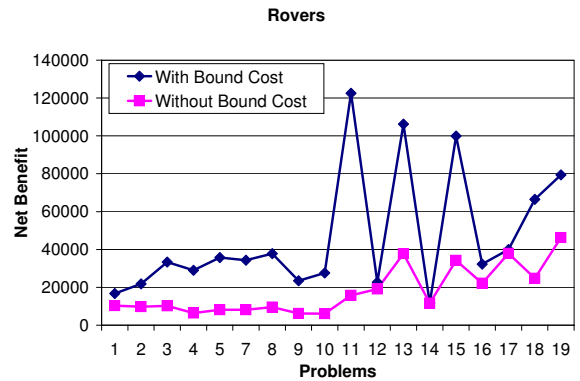

Figure 4: Comparison of utilities for our rovers domain

to represent numeric quantities for the number of packages at each location and the maximum capacity of each vehicle. Goals are then intervals of the number of packages we want at a particular location. For example, we may start with 10 packages at location l_1 and want to deliver between 5 and 7 packages to l_2 and at least 2 packages to l_3. Delivering a package to a certain location may be more beneficial than to the other (e.g. delivering 5 packages to l_2 gives more utility than delivering the same number of packages to l_3). The cost of delivering packages varies depending upon the distance between locations.

4.2 Results

All results were collected using a 2.67Ghz P4 machine with, 1 GB of RAM and 512 MB allocated to the planner. Given that $Sapa^{\mathcal{M}ps}$ employs an anytime search algorithm and continues to look for better quality plans as it is given more time, we set the time limit of 1200 seconds for all the tests[5]. We compare the final plan quality (i.e. the cumulative utility of goals achieved minus the cumulative action cost) on two heuristic approaches: (1) based on the cost-propagation as described in this paper; (2) only propagate the upper/lower bounds on numeric variables to check for numeric goal satisfaction, but do not propagate the goal achievement cost. The plan qualities are measured by the total achieved goal utility minus the total action cost. We were unable to make comparison with external planners because we are not aware of any other planner that is capable of solving the same problems.

Figure 4 shows the comparison results between two approaches listed above for the Rovers domain. The results clearly show that the cost-propagation helps $Sapa^{\mathcal{M}ps}$ return better quality solutions in all but 3 problems (with one of lower quality). The average improvement on the solution quality is 3.06 times better. The plans found are of reasonable size with a average/longest plan length in terms of number of actions is 30.5/77 (with cost propagation) and 20.9/67 (without cost propagation).

Figure 5 shows the results for the Logistics domain. Within the time limit, $Sapa^{\mathcal{M}ps}$ solves 19 of 20 problems with cost-propagation while only 10 (all with lower quality) without cost-propagation. Among the 10 problems that both approaches can solve, the average improvement in plan qual-

---

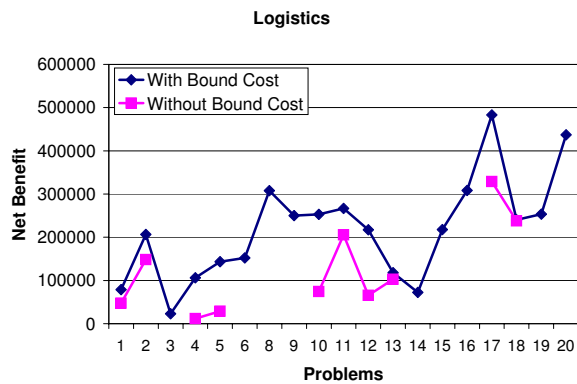[5] $Sapa^{\mathcal{M}ps}$ usually finds the first solution with positive benefit values in very short time

Logistics

Figure 5: Comparison of utilities for our logistics domain
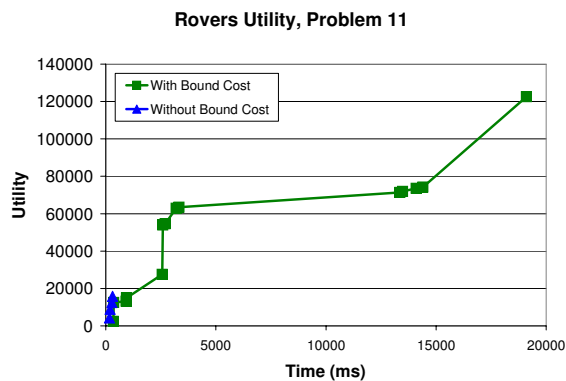
Rovers Utility, Problem 11

Figure 6: Plan net benefit as a function of time.

ity is 2.88 times. The average/longest plan length in terms of number of actions is 71.7/284 (with cost propagation) and 50.3/157 (without cost propagation) in this domain.

The results in these two domains confirm that the cost-propagation technique is indeed useful in helping $Sapa^{Mps}$ find better quality and larger plans, compared to only doing bound propagation.

Figure 6 shows how quickly the utility of a typical problem is found (in this case, problem instance 11 of rovers). Since we are using an anytime A* search framework, the utility increases gradually during search. The cost propagation heuristic over numeric bounds enables the planner to find greater utility plans.

5 Related Work

Our primary focus in this paper was to develop effective heuristics for handling numeric goals in the context of over-subscription planning. As we mentioned in the introduction, the recent interest in over-subscription planning has lead to the development of several algorithms [van den Briel *et al.*, 2004; Smith, 2004]. None of these approaches can handle numeric goals, and thus do not address the challenges involved in deciding the degree of satisfaction of a goal. Earlier work on the PYRRHUS planning system [Williamson & Hanks, 1994] did consider the issues of handling goals that allow different degrees of satisfaction in terms of achievement time. However, PYRRHUS did not have effective heuristics for directing its search.

As discussed in the paper, our work is also related to existing efforts on cost propagation over planning graphs (c.f. [Do & Kambhampati, 2003]) and propagating reachability information over numeric goals (c.f. [Sanchez & Mali, 2003; Hoffman, 2003]). Our method of propagating the upper and lower bounds of numeric variables over the RTPG has its roots in work done by Koehler in the Metric-IPP planner [1998].

6 Conclusion

Many real-world planning scenarios are over-subscribed, and require that the planner carefully balance the utility of the achieved goals and the cost of actions in the plan. In this paper we focused on extending over-subscription planning to handle numeric goals as well as a mix of hard and soft goals. Our primary technical contributions involved effective approaches for reasoning about the expected net benefit of a partial plan in the presence of numeric goals that allow different degrees of satisfaction (with noncomitant utilities). We have empirically demonstrated the effectiveness of our heuristics in leading the planner to solutions with higher net benefit. For the future, we are investigating extensions to our heuristic to handle delayed satisfaction of goals (i.e. goals whose utility depend upon achievement time).

References

[Blum & Furst, 1997] Blum, A., and Furst, M. 1997. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence* 90:281–300.

[Boutilier et. al., 1999] Boutilier, C., Dean, T., and Hanks, S. 1999. Decision-Theoretic Planning: Structural Assumptions and Computational Leverage. In *JAIR 11 (p.1-94)*

[Do & Kambhampati, 2003] Do, M. and Kambhampati, S. 2003. Sapa: a multi-objective metric temporal planer. In *JAIR 20 (p.155-194)*

[Do & Kambhampati, 2004] Do, M. and Kambhampati, S. 2004. Partial Satisfaction (Over-Subscription) Planning as Heuristic Search. In *Proc. of KBCS-2004.*

[Hoffman, 2003] Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. In *JAIR 20 (p.291-341)*

[Koehler, 1998] Koehler, J. 1998. Planning Under Resource Constraints. In *Proc. of ECAI-98 (p.489-493)*

[Sanchez & Mali, 2003] Sanchez, J. and Mali, A. 2003. S-MEP: A Planner for Numeric Goals. In *Proc. of ICTAI*

[Smith, 2003] Smith, D. 2003. The Mystery Talk. *Plannet Summer School*

[Smith, 2004] Smith, D. 2004. Choosing Objectives in Over-Subscription Planning. In *Proc. of ICAPS-04.*

[van den Briel *et al.*, 2004] van den Briel, M., Sanchez, R., Do, M, and Kambhampati, S. 2004. Effective Approaches for Partial Satisfaction (Over-Subscription) Planning. In *Proc. of AAAI-04*

[Williamson & Hanks, 1994] Williamson, M., and Hanks, S. 1994. Optimal Planning with a Goal-Directed Utility Model. In *Proc. of AIPS-94.*