

Choosing Objectives in Over-Subscription Planning

David E. Smith

NASA Ames Research Center
Mail stop 269-2
Moffett Field, CA 94035
desmith@arc.nasa.gov

Abstract

Many NASA planning problems are over-subscription problems – that is, there are a large number of possible goals of differing value, and the planning system must choose a subset that can be accomplished within the limited time and resources available. Examples include planning for telescopes like Hubble, SIRTF, and SOFIA; scheduling for the Deep Space Network; and planning science experiments for a Mars rover. Unfortunately, existing planning systems are not designed to deal with problems like this – they expect a well-defined conjunctive goal and terminate in failure unless the entire goal can be achieved. In this paper we develop techniques for over-subscription problems that assist a classical planner in choosing which goals to achieve, and the order in which to achieve them. These techniques use plan graph cost-estimation techniques to construct an orienteering problem, which is then used to provide heuristic advice on the goals and goal order that should be considered by a planner.

1. Introduction

Many NASA planning problems are over-subscription problems – that is, there are a large number of possible goals of differing value, and the planning system must choose a subset that can be accomplished within the limited time and resources available. For example, space and airborne telescopes, such as Hubble, SIRTF, and SOFIA, receive many more observation requests than can be accommodated. As a result, only a small subset of the desirable requests can be accomplished during any given planning horizon. For a Mars rover mission, there are many science targets that the planetary geologists would like to visit. However, the rover can only visit a few such targets in any given command cycle because of time and energy limitations, and limitations on the rover's ability to track targets.

Unfortunately, planning systems are generally not designed to deal with over-subscription problems. Most systems expect to be given a well-defined conjunctive goal and attempt to synthesize a plan to achieve the entire goal. They are not able to consider the values of the different goals, or to choose an appropriate subset of the goals that can be accomplished within the limited time and resources available.

In practice, most over-subscription problems have been addressed by using simple “greedy” approaches. For an

earth-observing satellite (where slewing is not possible, or slewing times are short) one can create a reasonable observation schedule by considering observations in descending order of their importance or priority. If the observation being considered is still possible, it is added to the schedule; otherwise it is discarded. This approach can work reasonably well for problems in which the cost of achieving an objective does not depend on the order in which the objectives are achieved. Unfortunately, this assumption does not hold for a Mars rover. The reason is that there is a significant cost in moving from one target to the next, and that cost depends on the distance and terrain between the two targets. In other words, the ordering of the targets has a strong influence on the overall cost of visiting those targets. As a result, much more powerful and informed search heuristics are required to help a planner choose the targets to visit and the order in which to visit them.

In this paper, we develop a technique for solving over-subscription planning problems. The technique involves constructing an abstracted version of the planning problem, and then using the resulting solution(s) to provide heuristic advice to the planner on the goals and steps that should be considered, and the order in which they should be considered. The abstracted version of the problem is formed by first estimating the costs of achieving each different objective (goal) using a plan graph. This information is then used to construct an *orienteering problem*, a variant of a traveling salesman problem. Solutions to the orienteering problem then provide the heuristic information needed for guiding the planner.

In the next section, we discuss plan graph distance estimation techniques, and show why they alone are not adequate for guiding search in over-subscription problems. In Section 3 we introduce the orienteering problem, and show how an orienteering problem coupled with plan graph distance estimation techniques can provide a useful abstraction of a rover planning problem. We then show how the solution to this orienteering problem can provide guidance to the planner search process. Throughout these two sections we limit our attention to a simple rover planning problem, where the mapping between the problem and the orienteering problem is relatively obvious. In Section 4, we generalize the graph construction and solution process so that it

applies to arbitrary planning problems. In Section 5, we present some preliminary experimental results on the rover problem. Finally we discuss related work and some current limitations of the approach.

2. Plan Graph Distance Estimates

A number of recent high-performance planning systems use a plan graph (Blum & Furst, 1997) to compute an estimate of the resources and time required to achieve goals from states encountered in the search process (e.g. Hoffman 2002, 2003; Do & Kambhampati 2003; Edelkamp 2003). This information is used to select among the different alternative search states.¹ To see how this works, consider the simple rover example shown in Figure 1, in which there are three

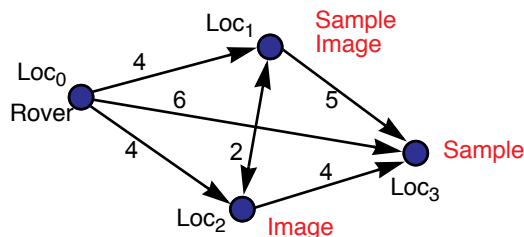


Figure 1: Simple rover scenario with three target locations. A sample and an image are desired at the first location, an image at the second, and a sample at the third. Only certain traversal paths are assumed possible because of terrain and visual tracking limitations. Numbers indicate time required to traverse paths.

target locations (rocks) with various paths along which the rover can travel. (Assume these paths are precomputed using path planning algorithms.) Both a sample and an image are desired at Loc₁, an image is desired at Loc₂ and a sample at Loc₃.

Figure 2 shows an abbreviated plan graph for the first two levels of this simple problem. Level 1 of the graph shows that it is possible for the rover to reach any of the three target locations with only one action. Level 2 shows that any individual experiment can be achieved with only two actions. Thus, the plan graph provides an optimistic assessment of which actions and propositions are possible. Since actions take time and resources, the graph can be used to compute estimates of the time and resources needed to achieve a goal or objective. In the example in Figure 2, numbers next to the actions indicate the cost of each action. With these numbers, we can use the graph to estimate the cost of achieving each of the objectives at level 2. For example, in order to have a sample at rock 1, we would need to move to rock 1 and then collect the sample, giving a cost of 4+3=7. These cost calculations can be done very rapidly in a plan graph by a simple forward sweep through the graph. Figure 3 shows a simple algorithm for doing this.

1. More precisely, these planning systems use this distance information to extract a relaxed plan for the goals, then use this relaxed plan as an estimate of the cost of achieving the goal from the search state.

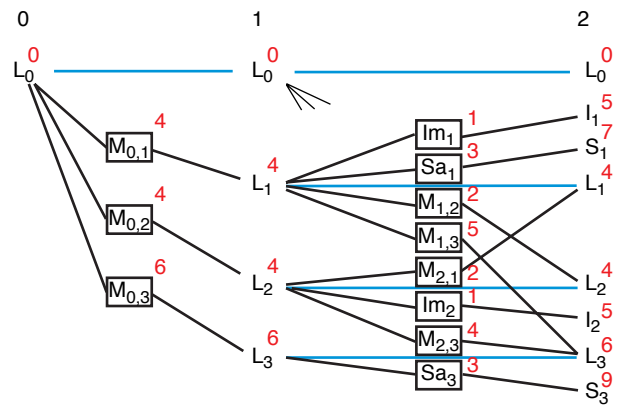


Figure 2: A portion of a simple plan graph for the rover problem. $M_{x,y}$ represents a move operation from x to y , Sa_x represents a sampling operation, and Im_x a close up image operation at location x . Numbers next to actions indicate action costs. Numbers next to propositions are the cost estimates for achieving those propositions. For simplicity, mutual exclusion relationships are not shown.

1. For all propositions p in the initial conditions (level 0), set $cost(p) = 0$.
2. For all other propositions p , set $cost(p) = \infty$
3. For each action a , set $cost(a) = \infty$
4. For level = 1, 2, 3, ...
 - a. For each action a at level, set:

$$cost(a) = c_a + \sum_{p \in preconditions(a)} cost(p)$$

- b. For each proposition p at level, set:

$$cost(p) = \min_{a \in support(p)} cost(a)$$

Figure 3: Simple plan graph cost estimation. Here c_a is the cost of an individual action a .

During planner search, this kind of heuristic “distance measure” can be used to select between different possible ways of achieving a goal. For example, if the goal is to have the sample at location 3, then it is better to go directly there rather than via either location 1 or location 2. If a direct path is not available, the graph would tell us that it is better go via location 2 (cost 8) rather than via location 1 (cost 9). It is this idea that provides much of the basis for the search guidance used by the competitive planners in the recent planning competition (Long & Fox, 2003).

For over-subscription problems, we could use the same strategy to estimate the cost of achieving each possible goal from the current state, and then try to use this information to select the most appropriate set of goals to achieve. However, we must also take the value of the goals into account. Thus the problem of choosing the set of goals becomes a sort of bin-packing problem in which we are trying to pack the most value into the bins of available resources. For example, suppose that the rover is at location 1, but has only four units of energy available. If we construct a plan graph starting at lo-

cation 1 and do the cost estimation, the graph will tell us that three goals are possible: Sample₁, Image₁, and Image₂, with costs of 3, 1 and 3 respectively. Solving the (trivial) bin-packing problem, we see that it is possible to achieve either Sample₁ & Image₁, or Image₁ & Image₂. If samples are worth more than images, then the first option is better. Otherwise, the second option is better.

While this approach works for this very simple example, it generally does not work well for the rover problem. The reason is that the cost of moving to a target depends heavily on the location of the previous target. Unfortunately, the heuristic distance estimates derived from a planning graph implicitly make the assumption that the goals are independent. For example, if the rover starts at location 0 with eight units of energy, the plan graph tells us that the three objectives are possible: Sample₁, Image₁, and Image₂ with costs 7, 5 and 5 respectively. Based on these costs we would be led to the conclusion that only one experiment can be performed while in reality it is possible to achieve Sample₁ & Image₁, or Image₁ & Image₂.

The problem is that in the plan graph in Figure 2, the cost estimate for getting to location 1 assumes we are coming from location 0, likewise for locations 2 and 3. However, locations 1 and 2 are close together, so the cost estimate of 4 for getting to location 2 no longer applies if we choose to go to location 1 first. Thus, the plan graph might allow us to pick the best or nearest single goal to accomplish, but it provides little immediate guidance if we want to visit several locations and achieve several goals in succession.

Several researchers have augmented plan graph cost estimation techniques to better account for interaction between actions in the graph. In particular, Hoffman (2002, 2003), Do & Kambhampati (2003), Edelkamp (2003) and Gerevini (2003) extract a relaxed plan from the planning graph and use this plan to estimate cost. However, this technique is primarily aimed at accounting for action duplication in estimating the cost of achieving a well-defined goal. For an over-subscription problem, this technique might improve the estimates for individual goals, but it does not solve the problem of interaction between the goals. The fundamental problem is that the resulting cost estimates assume that the goals are independent, and they are not. Thus, plan graph cost estimation alone does not seem to provide an adequate mechanism for choosing goals and goal ordering in such problems.

3. The Orienteering Problem

To overcome the difficulties mentioned above, we observe that there is a strong similarity between the rover problem and a variant of the traveling salesman problem known as an *orienteering problem* (Keller, 1989). In an orienteering problem, we are given a set of cities, a prize for each city (possibly zero), and a network of roads between cities. The objective is for a salesman to collect as much prize money as possible given a fixed amount of gas. The orienteering problem has been studied extensively in the operations research literature, and both exact and approximate algorithms have

been developed for solving this problem (Keller, 1989). To recast the rover problem as an orienteering problem, the cities become target sites, and the roads are paths between different targets, with costs corresponding to the resources required for the rover to traverse the path. The prizes are the scientific values of the experiments at a given target site. However, since there can be multiple experiments possible at a given site, and there are time and resource costs associated with each experiment, we need to create a separate “city” in the graph for each experiment at a site. We then add directed edges from the site to the experiments at that site, and return edges from the experiments back to the site. The resulting graph for our simple rover example is shown in Figure 4.

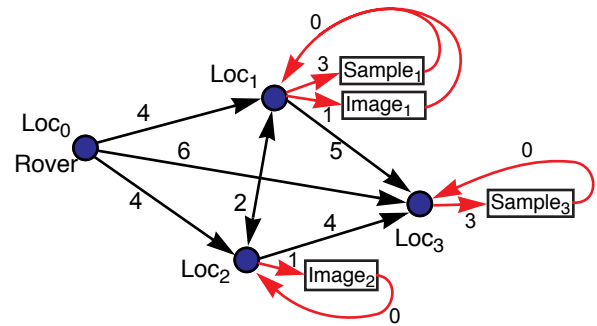


Figure 4: Orienteering graph for the rover problem. Cost estimates for each experiment are obtained using plan-graph cost estimation techniques.

We can assign a cost of zero to the return edges from an experiment to a site. However, an edge from a target site to an experiment should be assigned a cost that reflects the time and resources required to perform that experiment. Thus, for the sample at target1, we label the edge from target1 to sample1 with a cost of 3, corresponding to the cost of obtaining sample1, once we are at target1. In our simple example, each experiment is only a single step, so it is easy to come up with these numbers. In reality, experiments require many steps, and planning is required to generate these steps. We obtain the numbers for experiment edges in the orienteering problem using plan graph cost estimates, as described in the previous section. In particular, we ignore rover location in the plan graph by assigning a cost of zero to all locations. We can then compute cost estimates for all the objectives in the plan graph. These estimates provide the numbers that we need for each experiment edge in the orienteering problem. We can summarize the approach as follows:

1. Construct a plan graph and use it to estimate the time and resources required to perform the different experiments at each target site. (This can be accomplished for all sites simultaneously by assigning a cost of zero to all locations in the graph.)
2. Construct an orienteering problem, like that shown in Figure 4, using path planning to compute the edge cost for each move between sites, and using the estimates computed in Step 1 for the edge costs between a site node and the science experiments at that site.

- Solve the orienteering problem and use the resulting solution to guide the planning process.

A solution to this orienteering problem suggests which sites the rover should visit, the order in which to visit those sites, and which experiments should be performed at those sites. This information can then be used as heuristic guidance for a planner.

Searching for Plans

It might appear that solving the orienteering problem provides an exact solution to the rover’s planning problem, and that no additional planning would therefore be necessary. While this is true in our very simple example, it is not true in general. There are several reasons for this:

- Some of the steps for an individual experiment may interfere with each other. As a result, the experiment cost estimates obtained from the plan graph may be inaccurate.
- Different experiments can share steps or can interfere with each other. For example, obtaining a sample and taking a close up image both involve deploying the arm, so it is possible to share this step if both experiments are performed. This interaction is not modeled by our abstracted orienteering problem, which implicitly assumes that the only interaction between experiments results from the location of the rover.
- There may be time constraints on certain experiments (perhaps due to illumination constraints), or there may be required events (like communication) that must occur at specific times. These constraints are not reflected in the orienteering problem, so the resulting solution may be flawed.

As a result, the orienteering problem is only an abstraction of the real planning problem, and the solution to the orienteering problem may not prove to be a solution to the actual planning problem.

Suppose that we have a solution to the orienteering problem, and use it as heuristic advice to a planner to suggest the goals and the order in which to achieve those goals. When detailed planning is performed, the resulting plan could turn out to be much better or worse than that predicted by the heuristic estimates. In either case, it may be desirable to continue searching for a better plan. One possible approach is to search for another solution to the orienteering problem and try this as heuristic advice. Many of the algorithms developed for solving the orienteering problem (Keller 1989) are based on either local search, or branch and bound, and can therefore be adapted to provide a stream of solutions to the problem. A more ambitious possibility is to update the edge costs for the orienteering problem to reflect the actual values found in planning. One could then solve the orienteering problem again, and use the updated solution to continue guiding the planning process.

4. Generalizing to Multiple Interactions

Thus far, we have assumed that the only strong source of interaction between experiments is the location of the rover. This assumption allowed us to construct an orienteering problem in which the “cities” correspond to locations and experiments. While this is a far better model than that provided using only plan graph cost estimation techniques, it may not provide sufficient guidance for some problems. For example, suppose that a rover instrument has a significant warm up cost (energy), but once it is warm, it can be kept warm for additional experiments with little additional (energy) cost. In this case, there may be considerable advantage to performing a sequence of experiments at different sites using that instrument. This violates our assumption that “location” is the only rover attribute for which there is strong interaction between experiments. In order to fix this problem, we need to consider instrument-status, along with rover location in our creation of the orienteering problem. Specifically, the cities in the orienteering problem now become location/instrument-status pairs. In effect, we are now solving the orienteering problem on a projection of the state space for the rover – a projection onto location and instrument-status propositions.

For our simple rover problem suppose that the instrument is required for the two imaging operations, but not for collecting the samples. The graph would consist of two copies of the orienteering graph from Figure 4, one for the instrument off, and the other for the instrument on. The two graphs would be cross connected by the operations of turning the instrument on or off, as shown in Figure 5. Because the im-

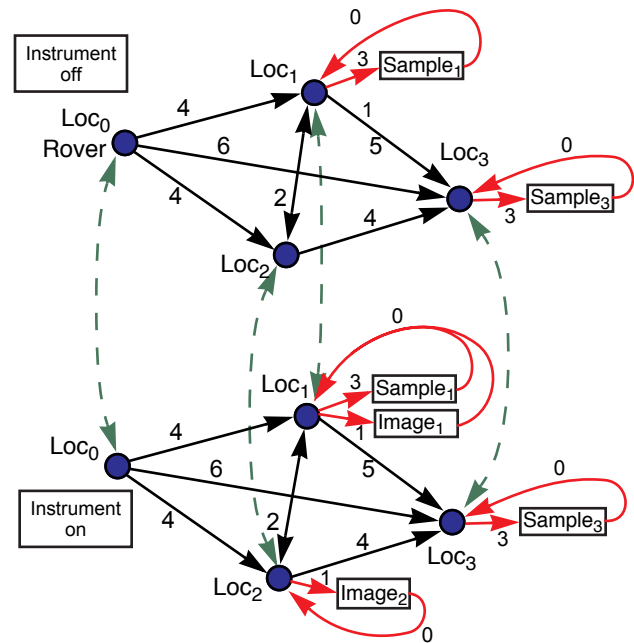


Figure 5: Orienteering problem for cross product of location and instrument status.

aging operations require that the instrument be on, these two

objectives only appear in the bottom half of the graph, where the instrument is on. However, the sampling operations don't rely on the instrument, so they appear in both the top and bottom halves of the graph.

This brings up an interesting issue: the graph in Figure 5 contains two copies of some of the objectives, $Sample_1$ and $Sample_2$. As a result, it is possible for the solution algorithm to collect the reward for this goal twice, by first visiting it in the upper part of the graph, then transitioning to the lower part (or vice versa). To fix this problem we add mutual exclusion (mutex) edges between all pairs of identical objectives appearing in the graph. We then modify the solution algorithm for the orienteering problem to respect those mutex constraints. This turns out to be fairly simple; the solution algorithm already keeps track of which cities have been visited so that it does not collect rewards twice when returning to a city. All that is necessary is that when a city is visited for the first time, we also add any mutex cities to the set of visited cities. This prevents the collection of any reward when visiting those cities.

For our simple rover problem, the structure of the graph in Figure 5 seems fairly obvious. However, if we wish to apply this technique more broadly, we need to be able construct this graph automatically. This turns out to involve several steps.

First, assume that we have a *basis set* O of propositions that will form the states in our orienteering problem. What we want to construct is the projection of the problem state space onto that set of propositions. However, this is generally intractable, since it requires that we enumerate the entire state space, project the states onto the desired propositions (get rid of the other propositions), and then combine identical projected states. Instead, we can construct an optimistic approximation of the projected state space by starting with a projection of the initial state, applying all applicable actions to this state, and projecting the resulting states. We repeat this process until no new projected states are found. This algorithm is summarized in Figure 6.

1. Let $I = \text{Initial state} \cap O$
(the initial state projected onto the propositions O .)
2. Let $States = \{I\}$, $Nodes = \{I\}$, $Edges = \{\}$
3. While $States$ is non-empty
 - a. Let $s = \text{Pop}(States)$
 - b. For each action a applicable in s :
 - Let $s' = \text{apply}(a, s) \cap O$
(the projection onto O of a applied in s)
 - Unless s' is in $Nodes$, add s' to $Nodes$ and $States$.
 - Unless $s' = s$, add the edge $\langle s, s' \rangle$ to $Edges$

Figure 6: Projected state space construction

For our example, the result of this process is the graph in Figure 5, but without the goal (experiment) nodes and edges.

Next, we need to add the goal (experiment) nodes and edges to the graph. To do this we need to figure out which goal (experiment) nodes connect to which projected states in

the orienteering graph. To determine this, we construct a relaxed plan for each goal in the plan graph. However, in doing this construction, we assume that all propositions in the orienteering graph are available in the initial conditions of the plan graph. In other words, in constructing the relaxed plan, we stop the backward search on a proposition if it has cost zero. The resulting relaxed plan will rely on a (possibly empty) set of "initial conditions" or zero-cost propositions belonging to the orienteering graph. This set of propositions corresponds to one or more of the states in the orienteering graph. As a result, we add a copy of the goal to the graph for each such state, and connect it to that state. In our rover example, the relaxed plan for $Image_1$ relies on the initial conditions Loc_1 and $Instrument-on$. As a result only one copy of the goal is added to the orienteering graph and is connected to the state $\{Loc_1, Instrument-on\}$. In contrast, the relaxed plan for $Sample_1$ relies only on Loc_1 so two copies are added, one for the projected state $\{Loc_1, Instrument-off\}$ and one for the projected state $\{Loc_1, Instrument-on\}$. Similar arguments apply to $Image_2$ and $Sample_3$.

An edge from a state node to a goal node is assigned a cost equal to the cost of the relaxed plan for that goal. Thus, for our example, the edges from the state node $\{Loc_1, Instrument-on\}$ to the goals $Image_1$ and $Sample_1$ are the cost of their respective relaxed plans (1 and 3). Finally, we mark any duplicate pairs of goals in the graph as mutex. This algorithm is summarized in Figure 7.

1. Construct a projected state space for the propositions in the basis set O
2. Mark all propositions in O as having zero cost in the plan graph
3. For each goal g :
 - a. Construct a relaxed plan r for g (halting at zero cost propositions in the plan graph)
 - b. Let f be the initial or *foundation* propositions for r
 - c. For each state s_i in the projected state space consistent with f , add a node g_i to the graph with reward equal to that of the goal g .
 - d. Add an edge from s_i to g_i with cost equal to the cost of the relaxed plan r . Add a return edge from g_i to s_i with cost 0.
 - e. Add a mutex edge between all nodes g_i .

Figure 7: Orienteering graph construction

In general, the size of the orienteering graph constructed by our method is exponential in the number of independent propositions. If O contains only location propositions, the size of the graph is on the order of the number of locations $|locations|$ (plus experiments). If we have location and instrument-state, it is $|locations| \times |instrument-states|$. If we have two instruments with significant warm-up costs, it is $|locations| \times |instrument-states|^2$, etc. Ultimately, if all propositions end up in the basis set O , the orienteering graph would be the complete state space for the problem. Thus the suc-

cess of this method depends on having a reasonably small basis set.

Identifying the Basis Set

For the rover problem, it seems fairly clear which attributes should be treated in the orienteering graph and which can be estimated using planning graph cost estimation. However, if we wish to apply these techniques to general planning problems, we need some way of automatically deciding which attributes belong in the orienteering graph. To do this, we can perform a kind of sensitivity analysis on the plan graph to find those attributes that have significant impact on the cost of achieving each goal. For example, in the plan graph shown in Figure 2, consider the relaxed plan for each one of the objectives. We note that all of these plans affect the location of the rover and leave it in a state other than the initial state. We therefore change the location in the initial conditions of the plan graph to see what impact this has on the cost estimates for different objectives. For the goal Sample₁, the estimated cost varies from 5 to infinity, depending on the initial location. As a result, location seems like a good candidate for treatment in the orienteering graph. Similarly, if the status of an instrument has a significant impact on the cost of achieving some of the objectives, it too would be a good candidate for the orienteering graph. Using this technique, we can automatically identify those attributes for which the goals strongly interact.

To make this precise, we first define the *net effects* of a (relaxed) plan as being the propositions that are true/false at the end of the plan, that were not true/false in the initial conditions. To construct the net effects of a plan, we simply begin with the initial state, and simulate the actions of the (relaxed) plan. At the end, we take the set difference between this “state” and the initial state. Note that in doing this simulation, we are ignoring action preconditions, so the fact that a relaxed plan may not be a legal plan has no impact on the computation of the set of net effects.² In the plan graph in Figure 2, the relaxed plan for achieving Sample₁ has the net effects {Loc₁, ¬Loc₀, Sample₁}. Similarly for the other objectives. Intuitively the net effects of a relaxed plan tell us how achieving that goal is likely to change the initial state.

Using this notion, the algorithm for constructing the basis set O is shown in Figure 8. This algorithm is somewhat more sophisticated than we suggested above. In particular, we do not actually change the initial conditions or structure of the graph when determining the effects of a relaxed plan on the cost of achieving other goals. Instead, we change the cost information in the graph by assuming, in turn, that the cost of each net effect is 0, and that propositions mutex with the net effect start out with infinite cost. For example, if we consider the net effect Loc₁ for Sample₁, we would set the cost of Loc₁ to 0 and the cost of Loc₀ to ∞, since Loc₀ is mutex with Loc₁. While this is much more efficient than changing the

1. Construct a plan graph for the problem and compute a cost estimate c for each goal g in the plan graph.
2. Let $O = \emptyset$
3. For each goal g in the plan graph:
 - a. Construct a relaxed plan r for g
 - b. For each net effect e of r :

Reset the cost information in the plan graph. (Initial conditions are set to 0, all other propositions and actions are set to ∞.)

Set the cost of e to 0 in the plan graph

For each proposition m mutex with e , set the cost of m to ∞.

Compute a revised cost estimate c' for each goal g' in the graph.

If $\exists g' \neq g$ for which c' is significantly different from c , add the effect e to O .

Figure 8: Constructing the basis set O for the orienteering graph.

structure of the graph, it is still important to note that this process is fundamentally n^2 in the number of goals, since we must examine the effect of each net effect for a goal on all the other goals.³

The last line of the algorithm in Figure 8 also leaves open the question of what we mean by “significantly different”. We might say that the cost of c' is significantly different from c if:

$$\frac{|c' - c|}{c} > \text{threshold}$$

However, this doesn't take into account the fact that some goals may be much more costly than others – large differences for one goal may be insignificant with respect to the overall planning problem. We could take this into account by instead making the cost difference relative to the total amount of resource available, or to the average cost over all goals, e.g.:

$$\frac{|c' - c|}{\text{available}} > \text{threshold}$$

How we set the threshold controls the size of the resulting basis set. If the threshold is small, then the basis set will be large and the number of states in the orienteering graph may explode. In the worst case, where all propositions end up in the basis set, the orienteering graph would simply be the complete state space for the problem. However, if we set the threshold larger, the basis set will tend to contain only those propositions that have a significant effect on the cost of achieving the different goals. Of course, it is important to realize that this method for automatically constructing the basis set is heuristic in nature. A relaxed plan for a goal is only

2. For a normal plan the set of net effects will always be consistent. Likewise for a relaxed plan if the initial state is fully specified. For our purposes it is not necessary that the set of net effects be consistent.

3. In practice there is often significant overlap between the net effects for different goals. As a result, caching can be used to further improve the above algorithm.

an approximation. It therefore may not accurately reflect the impact that achieving the goal will have on other goals. Furthermore, we are considering each net effect separately, which may exacerbate or disguise interactions.

5. Implementation and Experiments

Currently, we have only a partial implementation of the techniques presented in the previous sections. In particular, our implementation does not yet automatically construct the basis set for the orienteering graph. Furthermore, our automated construction of the orienteering graph currently only handles basis sets involving a single predicate, such as location. The implementation consists of:

1. Plan graph construction – a simple plan graph is constructed to quiescence for the planning problem.
2. Cost estimation – given a basis set of propositions to be ignored (such as location propositions), costs are computed for the objectives (goals) by setting the costs for ignored propositions to zero and performing the standard forward sweep through the plan graph.
3. Relaxed plan extraction – relaxed plans are extracted from the planning graph for each objective, assuming that all propositions with zero cost are available in the initial conditions.
4. Orienteering problem construction – given a single basis predicate (such as location), an orienteering problem is constructed corresponding to the projection of the state space onto that predicate. This projection is extracted from the propositions and set of transitions present in the plan graph for that predicate.
5. Goal node addition – each objective (goal) is added to the orienteering graph. It is connected to the projected state node in the orienteering graph corresponding to the zero-cost proposition used in the relaxed plan for that goal.
6. Orienteering problem solution – A beam search (using a greedy heuristic lookahead function for evaluation) is used to find solutions to the orienteering problem.
7. Planner guidance – the best solution to the orienteering problem is used to supply the goals to a POCL planner. The goals are fed to the planner one at a time in the order suggested by the solution to the orienteering problem. The planner can link to actions already in the plan structure, but cannot violate existing causal links. Planning terminates when resources are exhausted, or no remaining goals (from the solution to the orienteering problem) can be achieved.

We have performed some preliminary experiments with this system on rover problems involving 10, 25, 50, and 100 rocks randomly distributed in a 50x50 square. Between 1 and 3 experiments were available at each rock with experiment values chosen randomly in the range of 1 to 5. 75% of the n^2 paths between rocks were assumed to be traversable, with costs equal to the distance along the path. For the large problems, we gave the rover sufficient resources to allow it to visit approximately 10% of the rocks. For smaller prob-

lems we tried a range of resource values. We used a beam width of 25 when searching for solutions to the orienteering problem.

In all cases, construction and solution of the orienteering problem is very fast (0.3 seconds for the most difficult problems). Our technique for solving the orienteering problem is approximate, so we are not guaranteed to find the optimal solution. However, by performing experiments with very large beam width, we believe that we are obtaining optimal solutions for smaller problems and solutions within a few percent of optimal for the largest problems. Solution quality tends to drop off with a beam width of less than 15, and a beam width of greater than 50 slows down the solution process significantly. For these problems, a beam width of 25 seems to provide a good compromise between solution speed and solution quality.

We have also performed preliminary comparisons of the resulting plan quality for our approach against plan quality using greedy search strategies. We are typically getting plan quality improvements averaging from 10% to 30% depending on the density of goals. When the field is rich in goals, greedy approaches tend to work reasonably well. But when the goals are widely scattered, or there are many intermediate waypoints with no reward, the choice of goals and goal order can make a large difference in net reward. Figure 9

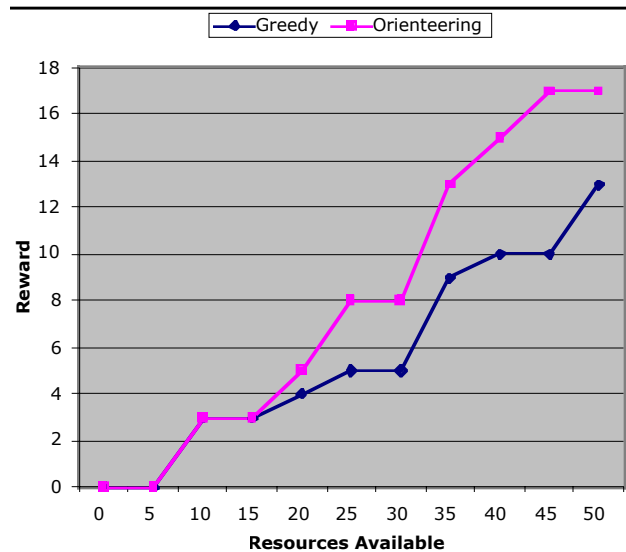


Figure 9: Plan quality (reward) for orienteering guidance and greedy guidance of a planner on a moderate sized problem involving 25 rocks in a 50x50 area.

shows a typical comparison of plan reward for the orienteering approach, and a fairly sophisticated greedy strategy as resources available to the planner range from 0 to 50. Eventually, as resources become plentiful, the greedy solutions catch up to the orienteering solutions.

6. Related Work

Few planning systems are able to solve over-subscription problems. Those that can are usually hand crafted for a spe-

cific domain and deal primarily with scheduling rather than planning problems. Examples of this include the scheduling systems for the Hubble space telescope (Kramer & Giuliano, 1997), for SIRTf (Kramer, 2000), and for the Landsat 7 satellite (Potter & Gasch 1998). Recently Kramer and Smith (2003) have investigated some heuristics for retracting tasks in over-subscription scheduling problems. However, it is not clear that these heuristics can address the kind of strong interactions found in the rover problem, or can be easily applied to planning problems. The Aspen system (Chien et. al, 2000) uses a local search approach to planning for over-subscription problems. However, it relies on simple greedy heuristics together with hand-coded domain-dependent search control information.

Markov Decision Processes (MDP) naturally permit the expression of multiple objectives and values for the objectives. Policy or value iteration can then be used to find optimal plans. However, it is tricky to prevent repeated collection of the same reward – in order to do this, one must add an additional proposition to the state for each possible goal. This increases the size of the state space by a factor of two for each possible goal.

Many recent planning systems make use of solutions to abstracted versions of the planning problem to guide planner search. These approaches typically extract relaxed plans from a planning graph, and use these relaxed plans to provide heuristic guidance to the planner (Do & Kambhampati, 2003; Edelkamp, 2003; Hoffman, 2002, 2003). The most sophisticated of these cost estimation methods are found in Metric-FF (Hoffman, 2003), which incorporates continuous variables into the planning graph, and SAPA (Do & Kambhampati, 2003), which considers time/cost tradeoffs in its heuristic calculations. Here we are constructing an orienteering problem to serve as the relaxed planning problem, rather than relying solely on a plan graph. As we argued in Section 2, a plan graph is not an adequate model for many over-subscription problems. However, we do continue to rely on plan graph cost-estimation techniques in order to seed the orienteering problem.

Estlin et. al. (2002) have used a TSP solver to help order locations and waypoints in a continuous planning and execution system for a rover. In this case, however, the mapping to the TSP is domain-specific, and is hard-wired into the system. The TSP also does not include cost or reward information for experiments at the different locations. Fox and Long (2001) use specialized algorithms like TSP solvers to provide both heuristic guidance and actual solutions to subproblems encountered during planning. In their case, they pick out specialized subproblems by recognizing their structural characteristics using domain analysis.

This is quite different from the approach taken here. In our case we form a single, global orienteering problem that is an approximation to the planning problem and use it for search guidance. The criteria for determining which propositions form the states in the orienteering problem is based on analysis of the independence between goals and on how the achievement of one goal affects the cost of achieving others.

7. Conclusions

In this paper, we developed a novel technique for solving over-subscription planning problems. The technique involves constructing an abstracted version of the planning problem and then using the resulting solution(s) to provide heuristic advice to the planner on the goals and steps that should be considered, and the order in which they should be considered. The abstracted version of the problem is formed by first estimating the costs of achieving each different objective (goal) using a plan graph. This information is then used to construct an orienteering problem. Solutions to the orienteering problem then provide the heuristic information needed for guiding the planner.

Although we presented the technique in the context of a rover problem, the technique applies to over-subscription problems more generally. In particular, the orienteering graph approach is useful whenever the order in which objectives are achieved has a strong influence on the cost of achieving those objectives. For such problems, simple greedy search strategies are not likely to work well.

The difficulty in applying this approach to general over-subscription planning problems is in recognizing which propositions should be part of the orienteering graph, and which can be treated using conventional plan graph cost estimation methods. We presented a technique for making these decisions automatically. It performs a kind of sensitivity analysis in the plan graph to determine how the cost of achieving each objective depends on the achievement of other objectives. If the cost of one or more objectives is highly sensitive to conditions that are changed when achieving other objectives, then those propositions are good candidates for the orienteering graph.

In Section 5, we indicated a number of limitations in our current, very preliminary implementation. We are in the process of removing these limitations. There are, however, some deeper issues that we have not yet addressed. As we mentioned in Section 3, time constraints on objectives, and on required activities such as communications, are not considered in the orienteering graph. Although there has been little work on solving orienteering problems with time constraints, it seems likely that some of the algorithms could be adapted to deal with them. This could further improve the accuracy of the approximate solutions, and thereby produce better search guidance for a planner. However, there is likely to be a cost to this increased accuracy, and it remains to be seen whether this additional accuracy will pay off.

8. Acknowledgments

Thanks to Maria Fox, Nicolas Meuleau and Sailesh Ramakrishnan for discussions on this topic. This research was supported by NASA Ames Research Center and the NASA Intelligent Systems program.

9. References

Blum, A., and Furst, M. 1997. Fast planning through planning graph analysis. *Artificial Intelligence* 90, 281–300.

- Chien, S., Knight, R., Stechert, A., Sherwood, R., & Rabideau, G. 2000. Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling for Autonomous Spacecraft. In *Proc. 5th Intl. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-00)*.
- Do, M. & Kambhampati, S. 2002. Planning graph-based heuristics for cost-sensitive temporal planning. In *Proc. 6th Intl. Conf. on AI Planning & Scheduling (AIPS-02)*.
- Do, M. & Kambhampati, S. 2003. Sapa: a scalable multi-objective metric temporal planner. *Journal of Artificial Intelligence Research* 20.
- Edelkamp, S. 2003. Taming numbers and durations in the model checking integrated planning system. *Journal of Artificial Intelligence Research* 20.
- Estlin, T., Fisher, F., Gaines, D., Chouinard, C., Schaffer, S. & Nesnas, I. 2002. Continuous planning and execution for an autonomous rover. In *Proc. 3rd Intl. NASA Workshop on Planning and Scheduling for Space*, Houston, TX.
- Fox, M. & Long, D. 2001. Hybrid STAN: Identifying and Managing Combinatorial Optimisation Sub-problems in Planning. In *Proc. of IJCAI-01*.
- Fox, M. & Long, D. 2003. PDDL 2.1: An extension to pddl for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20.
- Gerevini, A., Saetti, A. & Serina, I. 2003. Planning through stochastic local search and temporal action graphs in LPG. *Journal of Artificial Intelligence Research* 20.
- Hoffmann, J. and Nebel, B. 2001. The FF planning system: fast plan generation through heuristic search, *Journal of Artificial Intelligence Research* 14, 253–302.
- Hoffman, J. 2003. The Metric-FF planning system: translating “ignoring delete lists” to numeric state variables. *Journal of Artificial Intelligence Research* 20.
- Keller, C. 1989. Algorithms to solve the orienteering problem: a comparison, *European J. Operational Research* 41: 224–231.
- Kramer, L. 2000. Generating a Long Range Plan for a New Class of Astronomical Observatories. In *Proc. 2nd NASA Workshop on Planning and Scheduling for Space*.
- Kramer, L. & Giuliano, M. 1997. Reasoning About and Scheduling Linked HST Observations with Spike. In *Proc. International Workshop on Planning and Scheduling for Space Exploration and Science*, Oxnard, CA.
- Kramer, L. & Smith, S. 2003. Maximizing Flexibility: A Retraction Heuristic for Oversubscribed Scheduling Problems. In *Proc. of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)*.
- Long, D. & Fox, M. 2000. Extracting route-planning: first steps in automatic problem decomposition. In *Proc. Workshop on Analysing and Exploiting Domain Knowledge, AIPS 2000*, Breckenridge, Colorado.
- Long, D. & Fox, M. 2001 Multi-Processor Scheduling Problems in Planning. In *Proc. International Conference on AI (IC-AI'01)*, Las Vegas.
- Long, D. & Fox, M. 2002 Generic Types in Planning. In *Exploring AI in the New Millennium* (eds B. Nebel and G. Lakemeyer), Morgan Kaufmann.
- Long, D. & Fox, M. 2003. The 3rd International Planning Competition: results and analysis. *Journal of Artificial Intelligence Research* 20.
- Nguyen, X. and Kambhampati, S. 2000. Extracting effective and admissible heuristics from the planning graph. In *Proc. 17th National Conf. on AI (AAAI-00)*.
- Potter, W. & Gasch, J. 1998. A photo album of earth: Scheduling Landsat 7 mission daily activities. In *Proc. International Symposium on Space Mission Operations and Ground Data Systems*.