Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

SIMULATION-BASED PLANNING FOR PLANETARY ROVER EXPERIMENTS

David Joslin

CSSE Department
Seattle University
Seattle, WA 98122, U.S.A.

Jeremy Frank
Ari K. Jónsson
David E. Smith

Intelligent Systems Division
NASA Ames Research Center
Moffett Field, CA 94035, U.S.A.

ABSTRACT

Time and resource limitations mean that current Mars rovers (and any future planetary rovers) cannot hope to achieve every desirable scientific goal. We must therefore select and plan for a subset of the possible experiments, maximizing some utility metric. The use of simulation in planning is appealing because of its potential for representing complex, realistic details about the rover and its environment. We demonstrate a planning algorithm that performs high-level planning in a space of plan *strategies*, rather than actual plans. In the current implementation, candidate strategies are evaluated by a simple simulation, and a genetic algorithm is used to search for effective strategies. Preliminary results are encouraging, particularly the potential for modeling uncertainty about the time required to complete actions, and the ability to develop strategies that can deal with this uncertainty gracefully.

1 INTRODUCTION

In this paper we consider the problem of planning activities for a planetary rover, whose capabilities are based on the current and future rovers. The purpose of these rovers is to perform experiments, but high demand and limited resources make it impossible to perform all requested experiments. In classical Artificial Intelligence (AI) planning, known as conjunctive-goal planning, we are given some number of goals and a solution must satisfy all of those goals in order to be valid (Ghallab, Nau, and Traverso 2004). Instead we have an *oversubscribed* planning problem, and must select a subset of the goals (experiments) to achieve and plan the actions that will achieve those goals (Smith 2004, Joslin and Smith 2005).

Each experiment is assigned a utility value. The objective is to perform experiments that achieve the highest possible total utility. Experiments are subject to a variety of constraints. Some, such as sky and atmospheric measurements, can be performed at any location, while others are associated with a specific rock or other feature of the landscape and require that the rover first drive to a certain location. All operations, including driving, take time and consume energy. Energy is replenished by solar panels, up to the maximum capacity of the batteries, and cannot be consumed below a safety margin. Experiments produce data which is stored on-board the rover until it can be transmitted. On-board memory is also a limited resource.

Our previous efforts on this domain were based on Squeaky Wheel Optimization (SWO) (Joslin and Clements 1998), adapted for oversubscribed planning (Joslin and Smith 2005). In that work we search the space of permutations of the possible goals. A permutation is mapped to a solution by a polynomial-time greedy constructor. If the next goal in the permutation can be achieved, the necessary steps (possibly driving to a new location, then performing the experiment) are added to the plan at the earliest possible time. A permutation can be thought of as representing a prioritization of the goals because the earlier a goal appears in the permutation the earlier it is considered by the greedy constructor. If a goal cannot be achieved without violating constraints in the current plan, the goal is discarded and the constructor moves on to the next goal in the permutation. The total utility of the resulting plan is the quality assigned to the permutation from which the plan was derived. The SWO algorithm uses "directed mutation" to modify the permutation, and the algorithm continues in this fashion until a time limit is reached, returning the best plan found up to that point.

Here we extend that earlier work in several ways. We have expanded the size of the problems from planning one day of rover activities to ten days. We also take some first steps toward modeling the uncertainty that is inherent to the rover experiment planning problem (Bresina et al. 2002). For example, the time required to

drive from one location to another depends on terrain, soil consistency and other factors, some of which are not known at planning time. Longer driving times use more energy, leaving less energy for subsequent experiments. A plan that specifies every detail in advance may not work because of uncertainty about the conditions that will hold when a step in that plan is to be executed. For example, the Pathfinder rover spent 40-70% of its time idle due to failed plans and no on-board system to recover from plan failure (Bresina et al. 2002). Replanning by the rover itself may be impractical due to limited computational power. Some approaches attempt at planning time to anticipate and address potential issues that can arise from uncertainty, but such approaches often fail to handle complex problems.

1.1 Simulation and Planning

One of our goals is to explore options for using simulation techniques for planning. Simulation techniques offer the possibility of accurately representing the rover and its environment, including uncertainty about factors such as weather, uncertainty due to incomplete information, uncertainty about the effects of actions, etc. Although it is possible to generate plans using simplified models, and then validate or possibly repair those plans using a more accurate simulation, our objective is to incorporate simulation from the beginning of the planning process so that we avoid problems and inefficiencies that can result from a mismatch between highly simplified models and the actual environment and controller.

The idea of simulation-based planning is appealing, but it is also at odds with our earlier approach described above. The SWO algorithm added actions to the plan at the earliest possible time, possibly ahead of actions already in the plan. In other words, as is common to many AI planning algorithms, it is possible to decide that some action will be in the plan but postpone a decision about exactly when that action will be performed.

We can't simulate the result of performing an action at some undetermined future time. On the other hand, the ability to insert steps at the earliest possible time was critical to the success of that algorithm. Repeating the experiments from Joslin and Smith (2005) with a constructor that only adds steps to the end of the plan produced extremely poor results. This is not surprising. Consider a goal with a high utility that is subject to tight constraints, such as a temporal window on the time at which the experiment can be performed. When the constructor has flexibility about the ordering of plan steps, putting this difficult but high-utility goal early in the permutation means that the constructor commits to achieving that goal and ensuring that other decisions won't interfere. Other "easier" goals can then be satisfied by adding steps to the plan before and/or after the steps that achieve the high-priority goal. Here the goal permutation is serving to reflect something about the relative difficulty of achieving the various goals.

If we restrict the constructor to build a plan in temporal order, then the first goal in the permutation must be the first goal satisfied in the plan. This is convenient for simulating the effects of achieving that goal, but now the permutation cannot represent a prioritization of goals in the same sense as before, and we lose a crucial characteristic of the SWO algorithm. A successful strategy for planning activities must leave sufficient flexibility for the rover's on-board systems to react to uncertain outcomes, but simultaneously build good quality plans that account for uncertainty in execution.

1.2 Overview of New Algorithm

The SWO approach of searching in an abstract space of goal priorities has been successful in the past, and the use of simulation seems very promising for the sake of greater accuracy, particularly for representing uncertainty. Our objective in designing a new algorithm was to find an approach similar to SWO but amenable to the use of simulation.

Under the unrealistic assumption that we can ignore uncertainty, plan execution is simply the application of the plan exactly as it was generated. When we model uncertainty we cannot view plan execution so trivially. The actual steps performed must depend on the actual outcome of the execution of previous steps. For example, whether or not we can perform the next experiment immediately may depend on how long it took to drive to the current location. The decision about whether to drive to a location at all may need to depend on the probability of being able to arrive at that location in time to perform high-utility experiments.

We therefore replace the greedy constructor algorithm with a greedy execution algorithm. With SWO it was important that the constructor be a polynomial-time algorithm so that a sufficient number of iterations could be performed to have a good chance of converging on a solution of good quality. That consideration is magnified by computational limitations of the rover. It is important that any computationally-intensive planning be done remotely, so that the execution algorithm can be fast.

With SWO a permutation of goals represents some notion of prioritization relative to a greedy constructor. The constructor is used to produce a plan, which is then executed. Due to uncertainty, however, we do not want a fully detailed plan, and instead rely on the rover's on-board system to intelligently act in response to action

outcomes. We therefore replace the goal permutation with a "strategy" that is used to guide the execution algorithm.

Although many implementations of strategies (and the corresponding execution algorithms) would be possible, we start with a simple implementation that has two parts. A strategy has a an "approximate path" that constrains the path the rover follows while allowing it some flexibility to behave opportunistically based on the current world state. A strategy also has a set of weights that guide decisions about whether to perform experiments at the current rover location, and which experiments to perform, or whether to drive to a new location. The strategy implementation is described in more detail below.

The role of a strategy is to capture something about the "big picture" of the problem and problem domain, so that at execution time we can make reasonably intelligent decisions quickly. The computational investment is in finding effective strategies. For this we used a simple genetic algorithm.

We stress that these results are preliminary. The rover simulation was designed to be sufficiently realistic to be interesting, but can be substantially improved by including other important details. The representation of uncertainty is simplistic. The scope of the test problems is reasonable, with a planning horizon of ten days and plans typically achieving between 50 and 100 goals, but again the level of detail could be expanded. The implementation of a plan strategy was kept very simple and straightforward; many options for improving on this implementation remain to be explored. The search mechanism likewise was a very simple, generic genetic algorithm, with opportunities for domain-specific improvements deliberately avoided for this initial effort. Many improvements are possible.

The following section describes the problem in more detail, and Section 3 explains the current implementation. Experimental results are described in section 4, and the final sections discuss related work, and our conclusions and directions for further research.

2 PROBLEM DESCRIPTION

In our previous work (Joslin and Smith 2005) we defined problems intended to capture key characteristics of current and anticipated rover planning requirements. Here we expand the size of those problems by a factor of ten to make the problems more realistic. The planning horizon is increased from one day to ten, and the number of goals, map area, and so on are all increased proportionately.

Figure 1 shows a map for a typical problem. The number shown at each location is the total utility for all experiments that can only be performed at that location. Circles represent rocks, including a buffer area around each rock that the rover is required to avoid. Lines represent allowable paths of travel.

The map is square with an area representing 400 square meters times the number of days in the planning horizon, i.e., for the problems with a ten-day planning horizon the map has a total area of 4000 m^2 or roughly 63 meters on a side. Rocks are generated randomly, with radii (including buffer area) selected from a uniform distribution between 0.25 meters and 1.25 meters, until the percentage of the total area covered is approximately 15%. "Target" points are defined on the perimeter of rocks until there are approximately ten targets for each day in the planning horizon.

Navigation is restricted to a set of paths between target points. An allowable path must not intersect the circle defined for any rock. In addition, if any three paths form a narrow triangle, the long edge of the triangle is removed. We require that every target point be reachable from every other target point, because we don't want to generate problems that are effectively small and easy because only a subset of the targets can be reached.

Goals are then generated, intended to represent a typical mix of the types of experiments that scientists might request. The types of experiments that can be requested are shown in Table 1. The first column shows the number of requests of each type that are generated for a problem instance, scaled by the number of days in the planning horizon. The second column shows the utility range. The utility is 5^x where x is a random value selected with a uniform distribution over the indicated range. With a range of 0 to 4, for example, the actual utility values will be 1, 5, 25, 125 or 625. In current practice, experiments are grouped into categories with each category representing goals that are much more important than goals in the next lower category. The assignment of utility values is designed to reflect this qualitative valuation.

The next column shows the location constraints. Some goals, such as panoramic imaging, can be achieved at any location, while others are associated with a particular rock or geographic feature and must be performed at a certain location. Where the location is constrained to be at a target point, a target is selected randomly from the set of targets defined for the map.

Temporal constraints apply to some goals, as shown in the next column of the table. A "workday" window of eight hours is defined for each day, representing the time during which adequate sunlight is available. All operations must be performed within a workday window, but some operations are more tightly constrained than that. For example, an image of a geographic feature

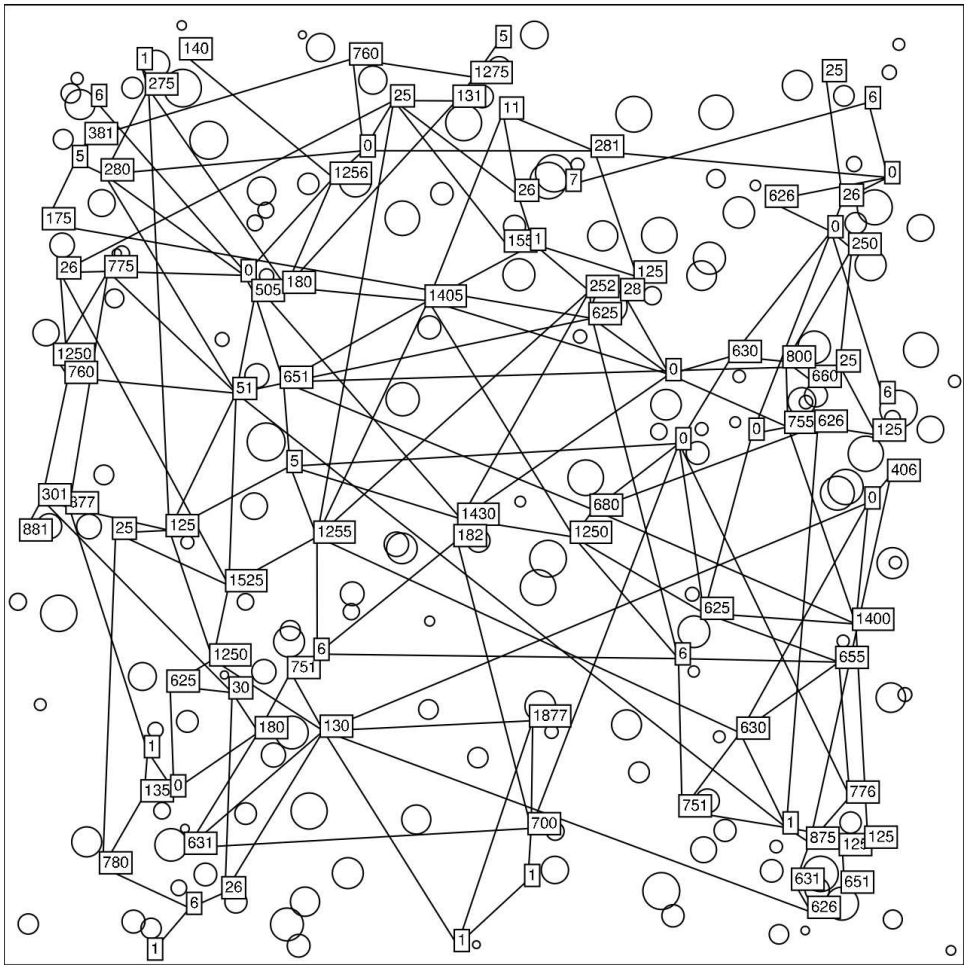

Figure 1: Map for Example Problem

may be desired with sunlight from a certain direction. Eight recurring temporal windows, with durations chosen randomly between 2 and 6 hours, were defined. An experiment that is constrained to occur within a particular window can be performed within that window on any day within the planning horizon.

The duration column shows the range of possible durations for each type of experiment. A duration is selected from a uniform distribution over that range. When we represent uncertainty, the actual value is chosen from a uniform distribution over a range defined by the duration assigned to the experiment plus or minus fifty percent. Drive time uncertainty is handled the same way, as discussed below.

The power column shows the rate at which energy is consumed during each type of experiment, in Watts. The energy consumption (in Watt-hours) is therefore the total duration multiplied by this value. The rover is constrained not to consume energy below a certain level. The battery is assumed to have a capacity of 600 Watt-hours, not including a safety margin that the rover should never plan to exceed. The initial energy is randomly chosen between 400 and 600 Watt-hours.

Energy is replenished each day by the solar collectors. A more accurate model would have the battery being recharged during the day, at a rate that varies with the time of day, and furthermore that rate should reflect some uncertainty because it depends on the positioning of the rover relative to the sun and other factors. We simplify this recharging by treating it as if it occurs instantaneously overnight, with up to 300 Watt-hours added, limited by the battery capacity.

The final column shows the data produced by the experiment. The data must be stored on-board the rover until it can be transmitted. We assume that up to 50 Mb of data are transmitted each day, and again simplify the representation by treating this as an instantaneous replenishment of storage capacity each night. The initial

Table 1: Goal Parameters

Goal type	**Goals per day**	**Utility**	**Location constraint**	**Temporal window**	**Duration**	**Power (Watts)**	**Data**
Camera	10-15	0-4	target	random, 2-6 hours	10-30 min	70	1 Mb/min
PanCam	15-20	0-4	any	random, 2-6 hours	10-60 min	70	1 Mb/min
MiniTES	10-15	0-4	50% target, 50% any	random, 2-6 hours	5-15 min	60	0.05 Mb/min
Microscopic Imager	5-10	0-4	target	None	5 min	65	0.5 Mb total
Mossbauer	2-3	2-4	target	None	5 hours	20	0.2Mb total

available capacity is chosen randomly from a range of 10 to 100 Mb.

In addition to the actions that the rover can perform for each type of experiment, the rover can drive to an adjacent location along one of the pre-defined paths. The duration of a drive (or the mean duration, if we are representing uncertainty) assumes an average rate of 5 meters per hour. The rover does not drive continuously because it must periodically stop and verify its location by using a camera to locate landmarks, and the average rate here reflects that mode of operation. Driving consumes 65 Watts of power. As with other operations it can only be performed during the workday window.

The current implementation does not allow for parallel actions. In the current Mars rovers, as well as in (Joslin and Smith 2005), a limited degree of parallel operation is possible. No experiments can be performed while driving.

3 IMPLEMENTATION

This section describes the current implementation. First we describe the implementation of strategies and the corresponding execution algorithm. We then discuss the implementation of the simulation that is used to evaluate a strategy. The last part of this section describes the genetic algorithm that is used to search for effective strategies.

3.1 Strategy and Execution

A strategy should represent high-level considerations of the problem and domain so that the execution algorithm can make reasonably intelligent decisions quickly. For the current implementation we represent a strategy in two parts: an "approximate path" and a set of weights that determine how the execution algorithm decides what action to perform next.

The approximate path is a sequence of map locations. The locations do not need to be adjacent in the graph representing the locations and allowed paths of travel. The execution algorithm is constrained to visit each of the locations in that sequence in order. Between those points, the algorithm is free to choose any edge in the graph that takes the rover closer to the next required location. Thus the sequence of locations exerts some control over the path of the rover while leaving some degree of flexibility for the execution algorithm. Where the locations are more closely spaced the strategy reduces the options available to the execution algorithm, and where they are more widely separated the strategy allows greater flexibility.

When the execution algorithm is invoked it decides what to do next. It can decide to begin performing an experiment at the current location, if all of the preconditions for performing that experiment are satisfied. It can decide to sit idle, which it might need to do, for example, to wait for the start of a temporal window required by an experiment. Or it can drive to another location. In order to drive, the preconditions must be met, i.e., sufficient energy and time must be available to complete the drive within the current workday window. Furthermore, the selected location must be adjacent in the graph of allowable paths, and the location must either be the next location in the strategy location sequence, or it must be closer (as measured by shortest-path distance) to that next location.

The strategy includes a set of weights that influence the way the execution algorithm makes these decisions. In this initial implementation we have tried to keep the strategy as simple as possible. There are three weights represented by non-negative floating-point numbers: the *idle weight*, the *local weight*, and the *threshold*.

We identify the goals that could be performed at the rover's current location, including goals that can only be performed at that location and goals that could be performed at any location. For each goal we determine

how long the rover would have to be idle before it could begin the experiment. It might be necessary for the rover to remain idle in order to wait for sufficient energy or data storage capacity to become available, or to wait for a temporal window, or some combination of these.

We score each of the goals as follows. The required idle time is multiplied by the *idle weight*, and then the duration for the goal is added. (For all of these calculations the "pessimistic" values for duration are used; see the discussion of the simulation implementation below.) The result is the adjusted duration. Larger idle weights provide a higher incentive to avoid sitting idle. In an environment where high-utility goals are plentiful, we would expect to see a large penalty for idle time. On the other hand, an environment in which high-utility goals are sparse might be better served by a low penalty for idle time, allowing the rover to wait at the current location in order to perform an experiment rather than moving on in the expectation of finding a more immediate opportunity later.

The goal utility is divided by the adjusted duration, giving the adjusted utility. This represents the rate at which utility is accumulated, taking any penalty for idle time into account. If the goal must be performed at the current location then the adjusted utility is the score for the goal. If the goal could be performed at any location, then the score is the adjusted utility multiplied by the *local weight*. Goals that can be performed at any location are more likely to be available later, after the rover has moved to another location. The *local weight* allows the strategy to give some degree of preference to goals that can only be performed at the current location.

Once all of the available goals have been scored, the execution algorithm ignores any goals with a score lower than the *threshold*. If any goals remain, the highest-scoring goal is selected for execution. If the experiment for that goal cannot begin execution immediately, the rover will remain idle until it can be executed. (Obviously there are a variety of simple optimizations that could be performed here, such as looking for lower-utility goals that can fill the idle time.)

If no goals have scores above the threshold, then the execution algorithm selects a destination to which the rover will drive. For each of the neighboring locations that are closer (according to the shortest-path distance) to the next required location in the strategy's approximate path, we sum the utility values for any unachieved goals that are constrained to be performed at that location. The next destination for the rover is the location with the highest total utility. This is a very limited view of the value associated with each possible location, but we rely on the strategy's approximate path to reflect higher-level considerations.

Currently these three weights control the rover's actions throughout plan execution, but finer-grained control would be possible. For example, we might have weights associated with each segment of the approximate path, so that the strategy could adapt to different properties in different areas of the map.

3.2 Simulation

When the execution algorithm has selected an action to perform (driving, or performing an experiment), the simulation selects a value for the duration of that action according to the associated probability distribution. Currently we use a uniform probability distribution for all simulated values. The duration thus selected is then used to calculate the energy consumption and the amount of data produced. Multiple simulation trials are used to estimate the expected utility for a strategy.

It would obviously be unfair to let the execution algorithm know what the simulated duration will be before a decision is made about what action to perform. When the execution algorithm is deciding what to do, it uses "worst case" values. The motivation for this is that operations are assumed to be non-interruptible, meaning that they cannot be suspended overnight, for example. This means that the execution algorithm cannot begin execution of an operation unless it can be completed during the current workday window under these "worst case" assumptions. Temporal constraints on operations are treated similarly, as are energy and data storage requirements. We do not consider the preconditions for an operation to be met unless they are met under these "worst case" assumptions.

By "worst case" we mean the worst case within operational parameters. We are not considering catastrophic failures of the rover, or unanticipated conditions that lead to operations failing or unusually long delays. Any operation that takes longer than the maximum specified for a given operation is viewed as a failure that requires review by ground control.

The current implementation was written in Python. A single simulation with a ten-day planning horizon takes almost one second of CPU time, with a typical plan satisfying 50-100 goals. The algorithm scales well because simulating the execution of a strategy is linear in the number of steps in the plan.

3.3 Search

We search the space of strategies using a simple genetic algorithm, and evaluate a strategy by simulating its execution. The initial pool consists of random weights selected from reasonable ranges, and location sequences generated by a simple greedy heuristic. For crossover

operations, two members are selected with a preference for higher fitness, and the child is created by pairwise-averaging the weights, and by selecting a crossover point in the location sequence and taking the sequence to the left of the crossover point from one parent, and to the right of the crossover point from the other. For random mutation, the weights are adjusted by a small random value, and each sequence location has a 10% probability of being replaced with a nearby location.

4 EXPERIMENTAL RESULTS

We generated twenty problems, with randomly-generated maps, goals, initial state, etc. Performance across all twenty problems was comparable, so the following results focus on the first problem in the set.

The first experiments used no uncertainty in the simulation. The duration for each operation is fixed at a typical value. Figure 2 shows the evolution of the strategy path after 50, 500 and 5000 generations of the genetic algorithm. The path becomes more streamlined, but at the same time the strategy parameters that control execution are being refined. The best strategy includes both the streamlined path and reduced parameters for the idle penalty and the threshold; as a result the rover tends to spend more time at each location, rather than moving on in order to (hopefully) achieve other higher-utility experiments.

For a simple test of simulation with uncertainty we defined the range of possible values to be a typical value plus or minus 50%, with values selected from a uniform distribution. In practice the probability distribution will not be so simple. For example, wheel slippage on the current Mars rovers, as a function of the slope of the terrain, is non-linear (JPL 2004), and other factors such as soil type will also have an effect on the time required to drive between two points. The use of uniform distributions is purely an implementation shortcut. Our approach could easily make use of complex distributions, and if desired, those distributions could apply different models for different regions of the map.

When the execution of an action is simulated, a value is selected randomly from the appropriate range. A strategy is evaluated by simulating its execution some number of times and and using the mean of the resulting scores. Figure 3 shows a histogram for the utility resulting from 100 simulations for the strategy that resulted in the best mean utility under uncertainty, after 200 generations of the genetic algorithm.

Figure 4 shows the evolution of strategy quality over time when the fitness function takes the mean value over 25 simulation trials. The dashed line shows the mean, and the error bars show the range of utility values for

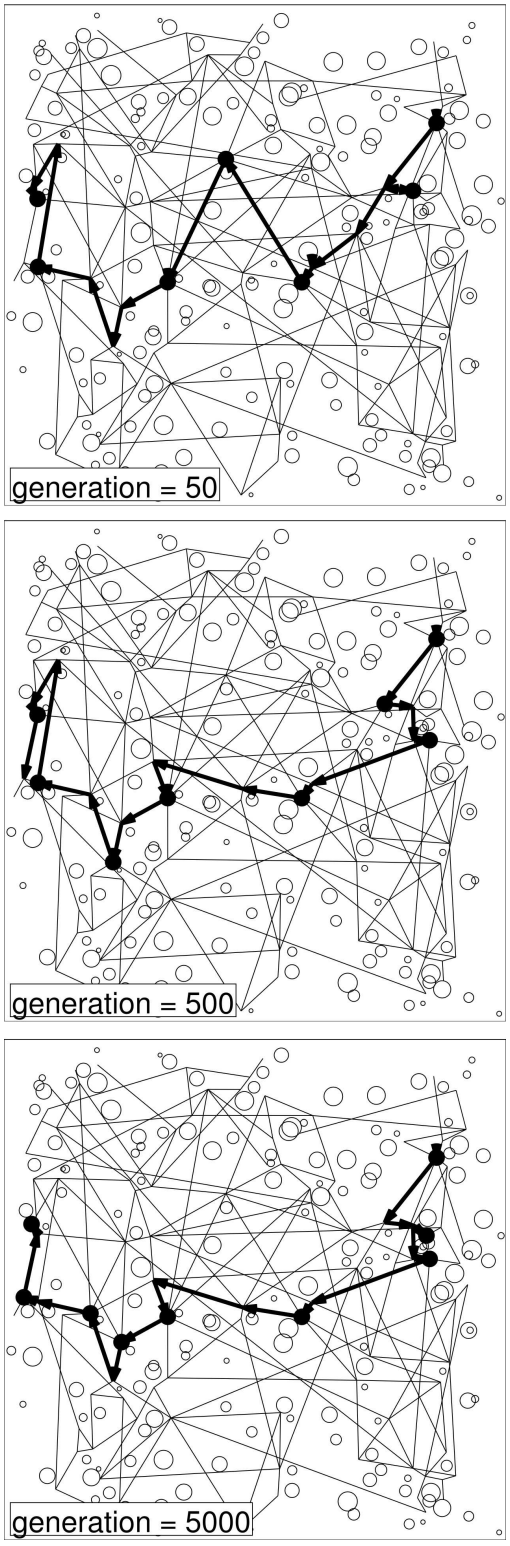

Figure 2: Plan Strategy Path Evolution (50, 500 and 5000 Generations)

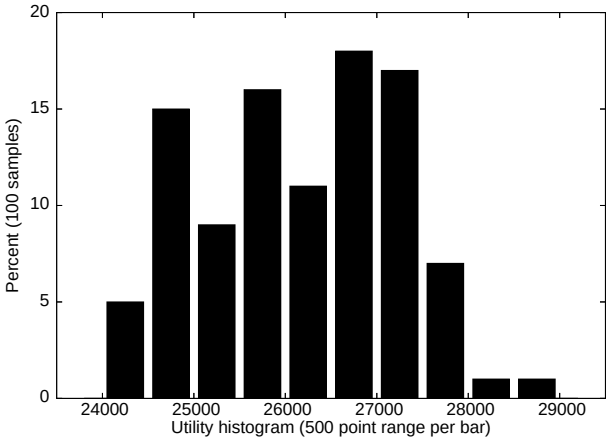

Figure 3: Utility Histogram over 100 Simulation Trials; Mean = 26432

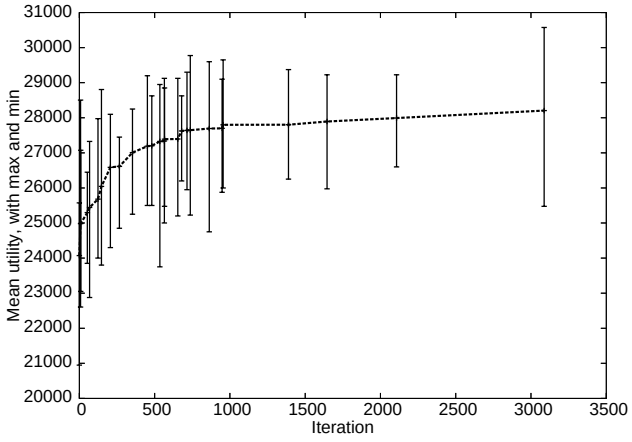

Figure 4: Evolution of Mean Utility

the 25 trials. For this experiment the algorithm was allowed to run for 5000 generations.

Note that the last improvement to the mean utility (generation 3087) also expanded the range between the minimum and maximum. This may be undesirable, but if so we can calculate the fitness value using some metric other than the estimated mean utility. We ran several experiments with different fitness calculations, with 5000 generations and 25 simulations per strategy. Simply minimizing the difference between the maximum and minimum utility turned out to be a bad idea. The algorithm quickly converged on a strategy that set the threshold very high, so that only a few very high-utility goals were ever attempted, and all could be achieved even under worst-case conditions. This reduced the uncertainty in the outcome to zero, but this is probably not a desirable mode of operation. Other experiments were done with the fitness calculated as the minimum and maximum utility over the set of simulations. These resulted in strategies that differed most noticeably in the value of the weights. With the minimum utility used as the fitness metric, the best strategy also diverged significantly from the path adopted by both of the other evaluation modes.

5 RELATED WORK

The current algorithm is based in part on ideas from SWO (Joslin and Clements 1998, Joslin and Smith 2005). In addition to the initial work on factory scheduling and graph coloring (Joslin and Clements 1998), SWO has been successfully applied to satellite downlink scheduling (Barbulescu, Whitley, and Howe 2004), satellite observation scheduling (Globus et al. 2004), project scheduling (Smith and Pyle 2004), and scheduling of airborne astronomical observations (Frank and Kürklü 2005), and other applications as well.

We can view the goal permutation in SWO as a form of strategy, although not one that is suitable for controlling an execution algorithm. Here we implement a strategy that might be considered more abstract, under which goal priorities are determined dynamically during execution, or during simulation of execution when we are searching for an effective strategy.

We deviate from the SWO approach of having a single goal permutation that is iteratively modified, and instead use a genetic algorithm to search for effective strategies. In this respect the approach we are taking is closer to that of Syswerda (1994). SWO can be viewed as a special case of this in which the population size is one, no crossover is used, and a collection of "directed mutations" are all applied at every iteration.

The new algorithm incorporates simulation in a fashion that allows us to represent and reason about uncertainty in the domain. The challenges of oversubscribed planning problems under uncertainty are outlined in (Bresina et al. 2002). As they discuss, many current approaches to planning under uncertainty do not apply to a domain such as the rover planning domain because, among other things, they assume that actions are instantaneous, and they do not allow for temporal constraints on actions.

Most of the approaches they discuss rely on building a plan with contingent branching. Some algorithms try to identify just the most important branch conditions. For example, in Gough, Fox, and Long (2004) conditional plans are developed that handle uncertainty about resource consumption by providing branches that can be taken if sufficient resources are available to make successful completion sufficiently probable. An algorithm for finding optimal contingent plans limited to k condi-

tional branches is demonstrated in (Meuleau and Smith 2003). As the authors point out, bandwidth limitations and other considerations provide strong incentives for keeping plans small. In our approach the information transmitted to the rover would not be a plan *per se*, but rather a strategy that guides the selection of goals and actions by an execution algorithm in the rover.

Simulation is well-suited to validation of plans. The MAPGEN system (Bresina et al. 2005), currently used to assist in activity planning for the current Mars rovers, includes simulators among its components. Specifically, EUROPA (Frank and Jónsson 2003) builds a plan, which is then evaluated in part by simulations of both power consumption (MMPAT) and the flight hardware sequence generator and checker (SEQGEN).

As noted earlier, our approach is an attempt to avoid problems that can arise from a mismatch between a simple planning model and a more realistic model used for validation by simulation, by instead using simulation during plan generation.

Simulation has also been used in Materials Requirements Planning (MRP) and Advanced Planning and Scheduling (APS) (Musselman, O'Reilly, and Duket 2002). Here the term "planning" is not used in exactly the same sense as in the field of AI, but there is substantial overlap. Orders and forecasts correspond to goals, and the problem may be oversubscribed in the sense that orders may have to be declined if manufacturing capacity is insufficient. Uncertainty exists in forecast quantities, delivery times, etc. Their system uses simulation to generate feasible schedules once orders have been assigned to workcenters and have been assigned start times. The scheduler uses a detailed simulation to assign low-level resources, to determine setup times (which may depend on how a machine was configured for the previous task), and so on. If some orders turn out to be infeasible to schedule the user may simulate various options such as hiring additional help or outsourcing, to try to find a better plan.

6 CONCLUSIONS

The key advantage of our approach to oversubscribed planning is the potential use of simulation during plan generation. Although the current implementation uses a very simple simulation, we anticipate being able to evaluate strategies and execution algorithms with much more sophisticated simulations. Simulations might, for example, consider weather predictions (temperature, wind levels, dust levels), incomplete knowledge about soil and terrain, and so on. Such factors can affect solar panel efficiency, drive times, available bandwidth, and other aspects of rover activity. For the sake of accuracy we may need to simulate dependencies between variables. If drive time is increased because poor visibility makes navigation more difficult, then solar panel efficiency is probably also degraded, and panoramic photography may be temporarily infeasible.

The key issue in applying this approach is whether or not a suitable strategy and execution algorithm can be devised. This is analogous to designing a suitable prioritization and constructor (and where applicable, the "directed mutations" that modify the prioritization) in Squeaky-Wheel Optimization. In the case of SWO, suitable implementations have been devised and found to be very effective in a wide variety of domains. For any new domain, however, there is no guarantee that a suitable implementation will be possible, for either algorithm.

Every element of our initial implementation could be improved. Additional parameters could allow for better fine-tuning of opportunistic behavior. The initial heuristic path generation could be much more sophisticated. A small amount of search for a good ordering of experiments that can be performed at the current location could easily reduce idle times. And numerous other enhancements would be possible as well.

Rather than tweaking this implementation, however, our near-term goal is to try to develop a better understanding of, and perhaps a generalization of, this approach of searching the space of plan strategies. As part of this, we hope to consider a wider range of strategy implementations. One intriguing idea is to use a sequence of goals with deadlines that define "critical commitments." The execution algorithm would be constrained to consider other goals only when successful completion of the critical commitments remains feasible under pessimistic assumptions where uncertainty is involved. Depending on actual outcomes during execution there may be more time, or less time, for opportunistic pursuit of secondary goals. Just as the approximate path in our current strategy serves to guide the rover in a certain direction, the requirement that a specific goal be achieved by a deadline will induce a preference for the rover moving in the direction of the required location for that goal (if any), and will impose constraints on resource consumption.

One appealing aspect of this alternate design for a strategy is that it provides a simple way for human experts to influence the plan, by setting critical commitments that the planner is required to include. The human planners could give the planning software as much or as little flexibility as desired. Another potentially appealing aspect of this design is that it is a conservative approach, in that the critical commitment goals set a lower bound on the utility, since all other decisions are made with those commitments having priority.

ACKNOWLEDGMENTS

The authors thank James Crawford for discussions of this work. This research was supported by NASA Ames Research Center, with funding from the NASA Intelligent Systems Program.

REFERENCES

Barbulescu, L., D. Whitley, and A. Howe. 2004. Leap before you look: An effective strategy in an oversubscribed scheduling problem. In *Proc. of the 19th National Conference on Artificial Intelligence*.

Bresina, J., R. Dearden, N. Meuleau, S. Ramakrishnan, D. Smith, and R. Washington. 2002. Planning under continuous time and resource uncertainty: A challenge for AI. In *Proc. of the 18th Conference on Uncertainty in Artificial Intelligence (UAI-02)*.

Bresina, J., A. Jónsson, P. Morris, and K. Rajan. 2005. Mixed-initiative planning in MAPGEN: Capabilities and shortcomings. In *Proc. of the ICAPS Workshop on Mixed-Initiative Planning and Scheduling*.

Frank, J., and A. Jónsson. 2003. Constraint-based attribute and interval planning. *Journal of Constraints Special Issue on Constraints and Planning* 8 (4): 339–364.

Frank, J., and E. Kürklü. 2005. Mixed discrete and continuous algorithms for scheduling airborne astronomy observations. In *Proc. of the 2nd Intl. Conference on Constraint Programming, Artificial Intelligence and Operations Research*.

Ghallab, M., D. Nau, and P. Traverso. 2004. *Automated planning: Theory and practice*. Morgan Kaufmann.

Globus, A., J. Crawford, J. Lohn, and A. Pryor. 2004. A comparison of techniques for scheduling earth observing satellites. In *Proc. of the 16th Conference on the Innovative Applications of Artificial Intelligence*.

Gough, J., M. Fox, and D. Long. 2004. Plan execution under resource consumption uncertainty. In *Proc. of the Workshop on Connecting Planning Theory with Practice at 13th Intl. Conference on Automated Planning and Scheduling (ICAPS'04)*, 24–29.

Joslin, D., and D. E. Smith. 2005. Squeaky-Wheel Optimization for planetary rover experiment planning. In *Proc. of the Intelligent Systems and Agents 2005 Conference (ISA2005)*.

Joslin, D. E., and D. P. Clements. 1998. Squeaky Wheel Optimization. In *Proc. of the 15th National Conference on Artificial Intelligence (AAAI-98), Madison, WI*, 340–346.

JPL 2004. Slip sliding away. `http://marsrovers.jpl.nasa.gov/spotlight/opportunity/b20_20040309.html`.

Meuleau, N., and D. Smith. 2003. Optimal limited contingency planning. In *Proc. of the 19th Conf. on Uncertainty in Artificial Intelligence (UAI-03)*.

Musselman, K., J. O'Reilly, and S. Duket. 2002. The role of simulation in advanced planning and scheduling. In *Proc. of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C. Chen, J. Snowdon, and J. Charnes, 1825–1830.

Smith, D. 2004. Choosing objectives in over-subscription planning. In *Proc. of the 14th Intl. Conf. on Automated Planning and Scheduling*.

Smith, T., and J. Pyle. 2004. An effective algorithm for project scheduling with arbitrary temporal constraints. In *Proc. of the 19th National Conference on Artificial Intelligence*.

Syswerda, G. 1994. Generation of schedules using a genetic procedure. U.S. Patent number 5,319,781.

AUTHOR BIOGRAPHIES

DAVID JOSLIN is an assistant professor in the Computer Science and Software Engineering department at Seattle University. He received his PhD from the University of Pittsburgh in 1996. His research interests include AI planning and scheduling, and game AI algorithms. His e-mail address is `<joslind@seattleu.edu>`.

JEREMY FRANK is a researcher in the Autonomous Systems and Robotics area at NASA Ames Research Center. He received his PhD. in Computer Science from the University of California, Davis, in 1997. His research interests are in automated planning and scheduling, with an emphasis on methods from AI, operations research and computer science. His e-mail address is `<frank@email.arc.nasa.gov>`.

ARI K. JÓNSSON is a senior research scientist with the Research Institute for Advanced Computer Science at NASA Ames Research Center. He received his Ph.D. in computer science from Stanford University in 1997, and has since worked on research and applications in the areas of constraint reasoning and automated planning and scheduling. He was the development lead for the MAPGEN mixed-initiative planning system, which is used to build daily activity plans for the MER rovers. His e-mail address is `<ajonsson@arc.nasa.gov>`.

DAVID E. SMITH is lead of the Planning and Scheduling Group at NASA Ames Research Center, serves on the Advisory Board for the Journal of Artificial Intelligence Research, and is Editor for special issues on the 3rd and 4th International Planning Competitions. Dr. Smith is a Fellow of the AAAI. His e-mail address is `<desmith@arc.nasa.gov>`.