Modular Optimization for Robotic Explorers

Kimberly Shillcutt, William Whittaker

Field Robotics Center
Carnegie Mellon University
Pittsburgh, PA 15213
kimberly@ri.cmu.edu

Abstract

Robotic explorers perform multiple tasks and have multiple constraints to optimize. Planning the best sequence of tasks requires information about how a given sequence will affect the constraints and goals of the explorer. Modules for all constraints and goals are created, each one of which evaluates a given plan from its point of view. The resulting values of the modules are then combined in a weighted sum, producing a score for each plan. The best plan is then selected.

Robotic Explorers

Robotic explorers are devices which investigate unknown environments. Typically, such robots are designed for places where humans either cannot go, or where sending humans would be too expensive or dangerous, such as space or remote earthly locations. As an extension of the human presence, robotic explorers should provide as much information as possible about the environment being explored, performing a wide variety of activities to act as a human's eyes and hands.

Robots have progressed from simple mechanical tools designed for single jobs to complex devices able to handle multiple tasks. These tasks may be independently carried out by separate parts and resources of the robot, be dependent on the right resources being available at the right time, or be dependent on the results of other tasks or user input. Such a robot must be able to plan the proper sequence of tasks.

One way to do this is to pre-program into the robot a well-defined sequence of tasks. A problem with this is that changes in the environment may require changes in the sequence of tasks. For a robot exploring an unknown environment, determining the proper sequence of tasks ahead of time may even be impossible.

Another way is to have the human operator send commands for the tasks one at a time. Problems with this occur when the robot is far away from the human operator, and time delays prevent seeing the results of a given command quickly. For tasks that depend on the state of the environment, such control from a distance is difficult, especially when rapid decisions have to be made.

A third way is to have the robot itself determine the sequence of tasks to be carried out, allowing changes in the environment to be incorporated into the planning. In this case, the robot reasons about its ability to perform various tasks as well as the probable results of performing those tasks, all with respect to the current environment. For a multi-purpose robotic explorer, this method of sequencing tasks needs to be used at least partially, with human operator decisions used for determining large-scale goals if desired.

A robotic explorer will have multiple tasks to perform, as well as multiple constraints to optimize. Some of these tasks might include covering an area as completely as possible, searching for new areas, investigating certain rocks or soil samples with additional sensors, extracting core samples with a drill, performing geological tests of rocks, or setting up equipment or structures for the use of future human explorers.

Constraints the robotic explorer may need to optimize include power gain from sources such as the sun or wind, power consumption, and visibility of a site. When traveling in a hilly area, the robot may need to constrain its path to avoid shadows as they move throughout the day, to maintain sunlight on its solar panels. The orientation of the robot, if it has fixed solar panels, is another factor which may have to be incorporated into the planning.

Task Planner

The task planner combines all the constraints with the desired tasks to generate the best plan for the robotic explorer. The specific constraints and tasks that are needed for a given robot will be determined ahead of time, as in the example application shown later. Each one of these constraints and tasks will be defined in one or more modules, as described in the section below. The following section explains how the task planner then integrates the outputs of these modules to select a plan.

Modules

A module is a self-contained unit which deals with a particular constraint or aspect of a task pertaining to the robotic explorer, either a cost or a benefit. For example, one module could be for solar power gained by the robot's solar panels, another could be for power consumed by the robot's locomotion system, another could be for information gained by the investigation of a rock with a spectrometer, and another could be for percentage of an area covered by the robot. These are all values of interest to the planner.

A constraint generally will be defined by a single module, such as the solar power gain constraint. The constraints generally need to be considered at all times, and their modules will be evaluated for all potential plans. When evaluated for a given plan, they produce a unitless value which reflects the cost or benefit to be produced when that plan is enacted. Costs will be weighted negatively, while benefits are weighted positively, as will be discussed in the following module integration section.

A task will generally be separated into multiple modules, as shown in Figure 1. For example, investigating a rock with a spectrometer can be separated into a module for the information gain of using the spectrometer on that rock, a module for the power consumption of deploying the spectrometer, and a module for the amount of time required (or the delay in resuming the previous plan being enacted). Such a task will also require the constraint modules to be evaluated, including the solar power gain module and the locomotion power consumption module. If additional costs or benefits are discovered to be beneficial to consider later, such as the cost of traveling and maneuvering to the rock from the current robot position, another module can be defined for that cost or benefit and easily added to the planner.

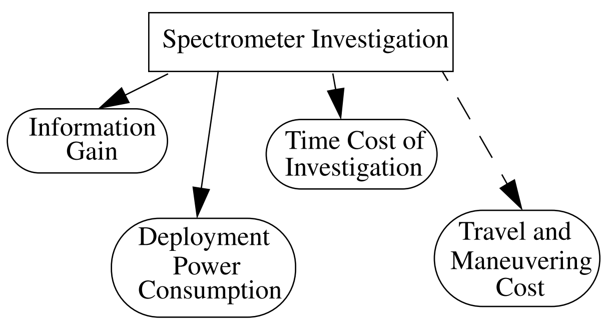

Figure 1. Task Decomposition.

The output of each module is unitless so that the values can be combined later into a total score for a given plan. Defining what the output should be, however, requires some insight into the particular constraint or task being defined by the module. Choosing the metric for a module, and choosing how to scale the resulting value, will affect what the planner decides to tell the robot to do.

The first step is to determine what modules are necessary for the robotic explorer and the environment in which it will operate. The configuration of the robot will determine some constraints, such as those that depend on the power sources available to the robot. The desired tasks also will determine the types of modules that are needed.

For a given task, such as covering an area, deciding to establish multiple modules depends on what aspects of the task are important. For instance, for the coverage task, information about what percentage of the area is covered and information about what percentage is covered multiple times may both be desired. In this case, two separate modules would be created for these pieces of information, and the output value for each would be designed accordingly. For these cases of percentage values, the unitless number to choose is simple, and can just be scaled between 0 and 1. For other modules, if a maximum value is known, scaling is again simple to do. For other cases, the scaling factor can be chosen somewhat arbitrarily, since weights for each module will have to be picked later anyway.

Many of the modules depend not only on the particular plan they are given, but also on the state of the environment at the time. Terrain will affect the robot's locomotion power consumption, as well as visibility constraints. The time of day and position of the robot will determine the location of the sun, which will in turn affect solar power gain. The modules must therefore have access to information about the current state of the environment, and be able to utilize this information in determining the costs or benefits of a given plan.

Module Integration

A plan consists of information such as the starting pose and location of the robot, the starting time of the plan, and the sequence of tasks to be carried out, such as covering a certain area or investigating first this rock and then another, with specific sensors. Given such a plan, each module takes the information it needs from that plan and produces its resulting cost or benefit value. If a given module is not relevant to that task, it will generate a NULL value.

The values from each module are then combined to produce a single output, with each module's value being multiplied by a weight before being summed together. The final output number is a score of how favorable that plan is. The plan which scores the highest is the one which will be selected by the task planner.

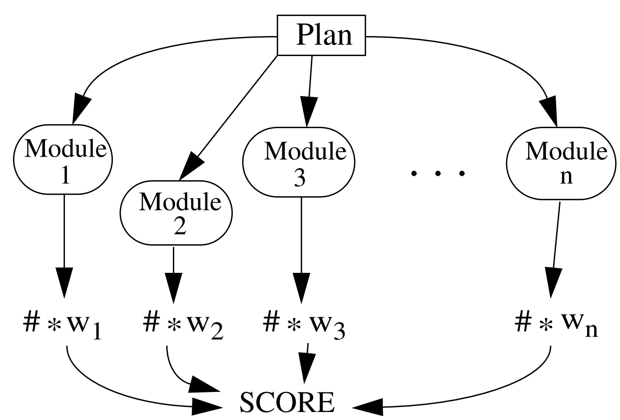

Figure 2. Plan Evaluation.

The weights can either be preset, determined ahead of time based on what constraints or tasks are more important, or can be set or changed by a user interface while the robot is in operation. Generally, costs will be given negative weights, and benefits will be given positive weights. In this way, each module produces a weighted vote for a given plan which the task planner then tabulates and uses to make its decision.

The task planner may give some constraint modules a threshold cutoff, where if the module's output value is below a certain number, the plan is automatically vetoed. This may be the case for a power gain module, for example, where if not enough power is produced, the robot cannot carry out the given plan.

Generating the various plans to submit to the modules is a difficult issue. Given a list of tasks which need to be accomplished, the task planner must suggest different plans which sequence the tasks in a certain order. One method would be to iterate all possible sequences, and test all the plans. For some situations, this may be too computationally expensive, and so a limited number of plans should be considered. Depending on the application for the robotic explorer, various heuristics could be applied to the tasks to generate potential sequences. An application-independent method of doing this has not been devised.

As the robotic explorer proceeds on its tasks, planning needs to be done concurrently to determine the next steps to take. Stopping to plan before acting is possible, but for the optimal use of time, concurrency is prefered. Separate processes for planning and executing can be run, utilizing the same plan data structure.

Replanning may also be needed due to differences between the plan and the actual performance of the robot. One heuristic to use to gauge when replanning is needed is to compare the actual result to the estimated result of the robot's location and time of completion of a task. If the difference is far enough off to affect the environmental parameters the modules use for calculating their costs and benefits, then replanning should occur. For example, the solar power gain depends on the location of the sun as compared to the robot, so a large enough change in the robot or sun's location can produce a large change in a given plan's solar power gain value. A threshold for the amount of difference required will have to be determined, however, for the given application.

Example Application

Robotic Antarctic Meteorite Search

A current project which may utilize this modular optimization planner is the Robotic Antarctic Meteorite Search (RAMS). This project involves a robotic explorer covering ice fields in Antarctica to search for meteorites. Humans currently perform this work, but the environment is cold and remote, and the job can be somewhat repetitive.

In the Antarctic summer, the sun is visible 24 hours a day, circling around the sky at a low angle above the horizon. Steady winds are also common, continually coming from the same direction. Thus, the sun and the wind are two potential power sources for a robotic explorer. Modules for the power gain from each of these sources will be needed if the robot is designed with solar panels and a wind turbine.

Another constraint which will require a module is the power required for locomotion. The power will vary depending on whether the robot is turning or going straight, traveling on ice or on snow, and perhaps on other issues.

The tasks this robot is expected to perform include covering an area as completely as possible and investigating potential meteorites with additional close-range sensors. These tasks need to be subdivided into multiple modules, reflecting the aspects of the tasks that are important to the planner.

Some of these aspects are the percentage of area covered, percentage of area covered more than once (or overlap, implying non-optimality), information gain of inspecting a potential meteorite with a particular sensor (increased ability to determine whether a rock is a meteorite or not), cost of deploying a particular sensor, computational cost of analyzing a sensor reading, and the time required to maneuver to a potential meteorite and take a sensor reading.

Sample Results

Preliminary work has been done on the solar and wind power gain constraint modules, as well as the modules for percentage of area covered and percentage of overlap. Several types of coverage pattern types were considered for which these four modules were then evaluated.

In this simplified case, the plan was simply to cover an area with a given pattern, starting at a point and continuing for 24 hours. The terrain was assumed to be completely flat, the sun clocked around steadily at the horizon for the entire 24 hours, the wind came from one direction at a steady speed, and the robot traveled at a constant speed.

The power sources included a four-sided, vertical solar panel square and a non-rotating wind turbine. Covering an area was defined as coming within a certain radius of the robot, so as to be visible in the robot's visual sensor. See Figure 3 for a sketch of this configuration. The actual numbers, such as solar panel area, wind turbine efficiency and visual sensor range, are not important to consider here, since only relative values are used in the planning process.

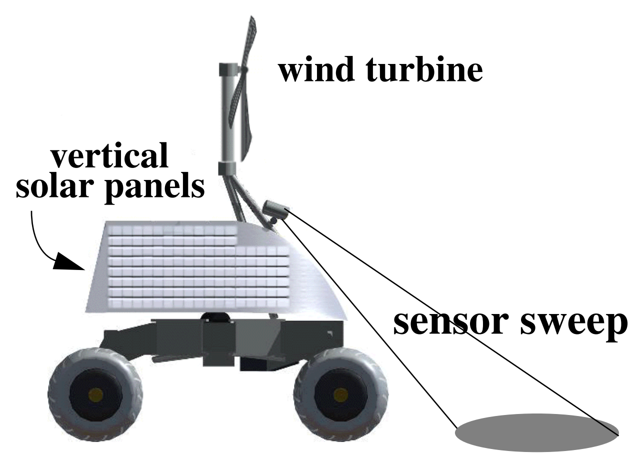

Figure 3. Primary planning components.

Four different pattern types were considered. The first was a pattern consisting of parallel rows of a given length, spaced apart the width of the robot's sensor sweep. The second was a spiral pattern, with the spirals spaced apart the sensor width. Third was a sun-following pattern, where the robot follows rows of a given length, but continually turns so that it maintains the same orientation with respect to the sun. This produces a spiral type of pattern, with a gap in the center, as shown in an overhead view in Figure 4. The position of the sun at various times during the day is labeled. The robot starts at 6 am, at the spot marked '*', when the sun is coming from the east, perpendicular to the robot's direction of travel. The exact pattern produced depends on the length of the rows traveled before the robot turns around. In this case, the length of rows was set to be the same for this pattern as for the first type of pattern.

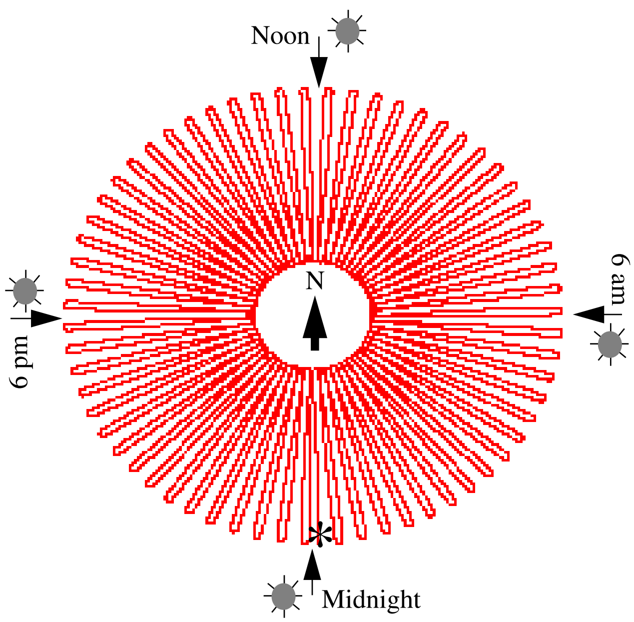

Figure 4. Sun-following pattern.

The fourth pattern type was a random pattern, where the robot follows a straight line segment of a random length, in a random direction. This type was meant to simulate the pattern a robot would follow if it kept deviating from a standard pattern to investigate potential meteorites, never maintaining a standard pattern for more than a short distance.

The amount of solar power gained over a 24 hour period was calculated for each pattern, as was the amount of wind power gained. The percentage of area covered was based on the minimum convex shaped bounding area that included all the points covered by the robot for a given pattern type. For the first pattern, this is just a rectangle, and for the second and third patterns, a circle. For the random pattern, the bounding area was also chosen to be a rectangle. The percent covered was defined as the total area covered divided by the bounding area. The percentage of overlap was calculated as the area covered more than once divided by the total area covered.

The module outputs for these four pattern type plans is shown in Table 1 below. Again, pattern 1 is rows, pattern 2 is spiral, pattern 3 is sun-following and pattern 4 is random. Based on the weights given each of these four modules, the planner would pick the plan with the highest score. For example, if solar power is given a weight of 10, wind power is given a weight of 2, percent covered is given a weight of 2, and percent of overlap is given a weight of -5, the four pattern types produce scores of 13.1, 12.4, 11.748, and 9.433 respectively. Pattern type one would therefore be selected. For weights of 20, 1, 5, and -1, the pattern type scores would be 24.2, 23.85, 24.438, and 19.129. In this case, the pattern selected would be pattern 3.

Pattern Type	1	2	3	4
Solar Power	0.91	0.91	1.0	0.91
Wind Power	1.0	0.65	0.65	0.64
% Covered	1.0	1.0	0.804	0.104
% Overlap	0.0	0.0	0.232	0.231

Table 1. Module Evaluations.

Related Work

Subdiving the complexities of a robotic explorer into modular components has been done in several different ways in the past. The mechanical design of a robot, based on modular mechanical components, was studied by Farritor et al. [1996]. Here, the authors propose using a hierarchical selection process and a genetic algorithm to select the best combination of generic, physical, modular components for a given task.

A slightly different approach was used by Sukhatme and Bekey to evaluate a given robotic explorer's performance. Based on a particular robot configuration, multiple criteria, similar to the task and constraint modules described in this paper, are used to evaluate how well the configuration meets the goals of the robotic explorer. These multiple criteria are not applied to a particular plan or task sequence, however.

Combining software modules to produce a plan is another type of modularization that has been studied. Farritor et al. describe how to use a genetic algorithm to combine various robot action modules to generate a plan for the robot to follow [1996 and 1997]. These software modules include basic actions such as "move this leg forward," or "use this sensor." A plan or task sequence is produced based on possible robot actions that can fulfill a given task.

Similarly, Gat et al. describe behavior control, where a planner arbitrates between behaviors which achieve certain tasks to determine which behaviors will generate the desired action of the robot. Modularization in the LAAS architecture [Alami et al.] comes from modules for elementary robot actions and computing tasks, which are then combined into a plan to accomplish a given goal. This paper, however, discusses the evaluation of multiple plans based on their usefulness in meeting various constraints or overall goals, rather than generating the steps which can be used to accomplish a single given plan.

Chatila et al. describe a mission planner which sequences tasks based on constraints such as time and other resources [1995]. This mission planner orders a set of tasks primarily based on time constraints. A task level planner later refines this sequence of tasks to optimize paths and trajectories or other characteristics. Different navigational modes, for example, can be selected based on the current terrain [Chatila and Lacroix, 1997]. Selection at the top level from among multiple plans and lists of tasks is not done, however.

This same mission planner is part of a broader architecture developed at LAAS, described in more detail in Alami et al. The planner considers both attributes of the environment and resources of the robot, and generates a partial plan for accomplishing a given set of tasks, satisfying temporal constraints. Refinements and completions of this plan are evaluated, and the version which least constrains the original partial plan is selected. These refinements deal with resolving resource conflicts, eliminating possibly inconsistent events or assertions, and adding subgoals that were not included in the original partial plan. Evaluation of constraint and task modules such as described in the current paper is not performed, however.

Conclusion

Modularization is important for a complex entity such as a robotic explorer. It enables a better understanding of the robot's components and how they fit together. It allows additions to be made easily and quickly. And it simplifies the creation of additional robotic explorers in the future, built to work in different environments and with different goals.

This paper has discussed the modularization of the planning process, allowing plans to be analyzed on the basis of the costs and benefits of importance to the robotic explorer. This method allows flexibility in selecting which goals are more important at the current time, by letting the weights for each module be changed, either by the user or a higher level planner. At different stages of the explorer's mission, different goals may be given the highest priority, but the modules used for calculating the best plan will remain the same.

Additional work remains to be done in generating the plans to analyze. For the example case of the meteorite-searching robot, the number and types of plans to be considered can be identified ahead of time, with the only variation being how many instances of the "examine rock" task are done at a time. However, when a larger number of tasks are possible, the planner needs a methodical means of ordering the possible sequences of steps to follow. Other planners exist which can produce such sequences, but determining the best plan generator for this method is still needed.

Eventually, this modular planner will be fully integrated into a robot desgned to explore a new environment, such as another planet. Building on other work in reliable navigation and sensor processing, such a robot can interact confidently with its environment. Programmed with a list of goals and priorities, the explorer will use its sensors to gain information about the environment and plan the best way to acheive its goals in that environment. The robot's priorities can change as more information is obtained, or as different stages of the mission are reached. Its findings will be sent back to Earth from time to time, but the robot will not necessarily need to receive commands from any human operators. It will be a completely autonomous agent, able to make decisions much as a human does, weighing the costs and benefits of different plans of action, and selecting the best means to accomplish its goals.

References

Alami, R., R. Chatila, S. Fleury, M. Ghallab, F. Ingrand, "An Architecture for Autonomy," *International Journal of Robotics Research*, Vol. 17, No. 4, pp. 315-337, April 1998.

Chatila, Raja, Simeon Lacroix, Thierry Simeon and Matthieu Herrb, "Planetary Exploration by a Mobile Robot: Mission Teleprogramming and Autonomous Navigation," *Autonomous Robots*, Vol. 2, pp. 333-344, 1995.

Chatila, Raja and Simon Lacroix, "A Case Study in Machine Intelligence: Adaptive Autonomous Space Rovers," *Proceedings of the International Conference on Field and Service Robotics*, Canberra, Australia, December 1997.

Farritor, S., Dubowsky, S., Rutman, N., and Cole, J. "A Systems Level Modular Design Approach to Field Robotics," *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, MN, April 1996.

Farritor, S. and Dubowsky, S. "A Genetic Algorithm-Based Navigation and Planning Methodology for Planetary Robotic Exploration." *Proceedings of the American Nuclear Society Sixth Topical Meeting on Robotics and Remote Systems*, Augusta, GA, April 1997.

Gat. E., Desai, R., Ivlev, R., Loch, J., Miller, D., "Behavior Control of Robotic Exploration of Planetary Surfaces," *IEEE Transactions of Robotics and Automation*, Vol. 10, No. 4, August 1994.

Sukhatme, Gaurav S. and George A. Bekey, "Multicriteria Evaluation of a Planetary Rover", *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Minneapolis, MN, April 1996.