Using Continuous Planning Techniques to Coordinate Multiple Rovers

Tara Estlin, Gregg Rabideau, Darren Mutz, and **Steve Chien**
Jet Propulsion Laboratory
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
{firstname.lastname}@jpl.nasa.gov

Abstract

This paper describes a dynamic planning system for coordinating multiple rovers in collecting planetary surface data. A distributed planning system is shown to generate rover plans for achieving science goals, coordinate activities among rovers, monitor plan execution, and perform re-planning when necessary. Specifically, we describe how rover command generation can be automated to help relieve some of the burden on human operators. We describe the issues inherent in planning for a distributed set of rovers and discuss how these issues can be addressed in a dynamic and uncertain environment. Finally, we describe a prototype system for automatically generating low-level commands and monitoring their execution for a team of rovers with the overall goal of achieving a set of geology-related science requests.

1 Introduction

Landmark events have recently taken place in the areas of space exploration and planetary rovers. The Mars Pathfinder mission was a major success, not only demonstrating the feasibility of sending rovers to other planets, but displaying the significance of such missions to the scientific community. Future missions are being planned to send additional robotic vehicles to Mars as well as to the outer planets and an asteroid [JPL, 1999]. In order to increase science return and enable certain types of science activities, future missions will require larger sets of rovers to gather the desired data. These rovers will need to behave in a coordinated fashion where each rover accomplishes a subset of the overall mission goals and shares any acquired information. In addition, it is desirable to have highly autonomous rovers that require little communication with scientists and engineers on earth to perform their tasks. An autonomous rover will be able to make decisions as how to best achieve science goals as well as being able to react to its environment and handle unforeseen events while achieving these goals.

An autonomous rover (or team of rovers) must respond in a timely fashion to a dynamic and unpredictable environment. Rover plans must often be modified in the case of fortuitous events such as science observations completing early and setbacks such as traverses taking longer than expected or hardware failures. We call this situation *continuous planning*, where a plan must be continually updated in light of changing operating context. In such an operations mode, a planner would be continuously updating the plan (e.g., every few seconds) based on sensor and other feedback, and then modifying the current plan accordingly to accommodate any new data. Making the planner capable of such timely responses has a number of benefits:

- The planner can be more responsive to unexpected changes in the environment. These changes could involve the status of executing activities, as well as updates to state (e.g., temperature, sun angle) and resource values (e.g., battery power).
- The planner can reduce reliance on predictive models since it will be updating its plans continually. Thus, inevitable modeling errors or uncertainties in the environment can be handled without causing plan failure and without explicitly representing all contingencies in the planning model.
- Rover fault-protection and execution layers need worry about controlling the rover over a shorter time horizon since the planner will replan within a shorter time span.

In a traditional cycle of plan, sense and act, planning is considered a batch process and the system operates on a relatively long-term planning horizon. For example, operations for Sojourner were planned on the ground on a daily basis [Mishkin *et al.*, 1998]. In this mode of operations, the rover state at the start of the planning horizon was pre-determined based on feedback from the previous day's operations. The science and engineering operations goals were then be considered, and a plan for achieving the goals would be generated. This plan or sequence was then uplinked to the rover for execution where it would be executed onboard the rover with minimal amounts of flexibility. If an unexpected event or failure occurred the rover would often be taken into a safe state by fault protection software. The rover would then wait in this state until the ground operations team could respond and determine a new plan. Correspondingly, if an unpredictable fortuitous event occurred, the plan could not be modified to take advantage of the situation. This paper presents a continuous planning approach to rover operations, which is intended to achieve a higher level of responsiveness in situations where replanning is required or beneficial.

Specifically we present work on using the CASPER (Continuous Activity Scheduling, Planning Execution and Replanning) [Chien *et al.*, 1999] planning system to control a set of distributed rovers for planetary operations. Based on an input set of science goals and each rover's initial conditions, CASPER generates a sequence of activities that satisfies the goals while obeying each of the rover's resource constraints and operations rules. Plans are produced by using an "iterative repair" algorithm which classifies conflicts and resolves them individually by performing one or more plan modifications. Once a valid command sequence is generated, commands are relayed to the rover's low-level control software for execution. Execution updates are relayed back from this software where they are monitored by CASPER. As information is acquired regarding command status and actual resource utilization, the planner can update future-plan projections. From these updates, new conflicts and/or opportunities may arise, requiring the planner to replan in order to accommodate the unexpected events. Planning activities are also distributed among the individual rovers where each rover is responsible for planning for its own activities. A central (non-dynamic) planner is responsible for dividing up the goals among the individual rovers in a fashion that minimizes the total time spent traversing by all rovers.

The planning system described in this paper has been integrated with a number of other software components to form a multi-rover execution architecture [Estlin *et al.*, 1999a; 1999b]. These components include: a machine-learning science analysis tool which analyzes planetary data and generates a set of goals for new science observations, a simulation environment that models multiple rover-science operations in a Mars-like terrain, a real-time multi-rover hardware and kinematics simulator, and control software from the NASA JPL Rocky 7 rover. This software is currently being tested on a geology-related science task where it must autonomously evaluate what science observations to perform, generate the necessary steps, and ensure execution of these steps is successful.

The remainder of this paper is organized in the following manner. We begin by characterizing the multiple-rovers application domain and describing our particular science scenario. Next, we present a multi-rover execution architecture which controls and coordinates operations for a team of rovers. We then focus on the planning aspects of this architecture; in particular we discuss our approach to distributed planning, our utilization of the CASPER continuous planning system, and our approach to plan optimization for this domain. The final sections discuss related and future work including system extensions and testing on real rovers.

2 Cooperating Rovers for Science

Utilizing multiple rovers on planetary science missions has many advantages:

- Multiple rovers can collect more data than a single rover. A team of rovers can cover a larger area in a shorter time where science gathering tasks are allocated over the team.
- Multiple rovers can perform tasks that otherwise would not be possible using a single rover. For instance, rovers landed at different locations can cover areas with impassable boundaries that would be unreachable by a single rover. Also, with several rovers, one rover can afford to take more risk and thus attempt tasks that usually might be be avoided.
- More complicated cooperative tasks can be accomplished, such as taking a wide baseline stereo image (which requires two cameras separated by a certain distance).
- Multiple rovers can enhance mission success through increased system redundancy. If one rover fails, then its tasks could be quickly taken over by another rover, helping to ensure mission success.

In all cases, the rovers should behave in a coordinated fashion, dividing goals appropriately among the team and sharing acquired information. In addition, it is desirable to have rovers behave in a dynamic fashion so that plans can be adjusted when unexpected events or failures occur.

Coordinating multiple distributed agents for a mission to Mars or other planet introduces some interesting new challenges for the supporting technology. Issues arise concerning communication, control and individual on-board capabilities. Many of these design decisions are related, and all of them have an impact on any onboard technologies used for the mission. For example, the amount of communication available will determine how much plan data can be easily shared among rovers to perform necessary re-planning. It also affects how much each rover can coordinate with other rovers to perform tasks. The control scheme will determine which rovers execute what tasks. For instance, some rovers may be utilized only for science data gathering, while other may be used for planning and/or science analysis. Decisions on the on-board capabilities of each rover can also determine the independence of a rover. Planning, execution and plan monitoring can be performed onboard all rovers or on a select few which will provide these capabilities as a service to other rovers.

For the framework discussed in this paper, we have initially chosen the configuration of a team of three rovers where each rover has a planning and science analysis tool on-board. Each rover can thus plan for its assigned goals, collect the required data, and perform science analysis on-board which will direct its future goals. In addition, each rover can monitor its own plan execution and perform re-planning when necessary. Central planner and science-analysis modules are assumed to be located on either a lander or one of the rovers, which are used to coordinate goals and science data.

Currently, we are evaluating our framework by testing its ability to build a model of the distribution of terrain rocks, classified according to composition as measured by a boresighted spectrometer. To perform testing for different planetary terrain models in a simulated environment, different rock fields (i.e., landscapes) are generated by using distributions over rock types, sizes, and locations. Science goals consist of requests to take spectral measurements at certain locations or regions. These goals can be prioritized so that, if necessary, low priority goals can be preempted (e.g., due to resource constraints such as low battery power).

Science goals are divided among the three rovers. Each rover is identical and is assumed to have a spectrometer on-board as well as other resources including a drive motor, a solar panel that provides power for rover activities, and a battery that provides backup power when solar power is not available. The battery can also be recharged using the solar panel when possible. Collected science data is immediately transmitted to the lander where it is stored in memory. The lander has a limited amount of memory and can only receive transmissions from one rover at a time. The lander can also upload data (and simultaneously free up memory) to an orbiter whenever the orbiter is in communication contact.

3 Multi-Rover System Architecture

The distributed planning system described in this paper is part of a multi-rover execution architecture that coordinates multi-rover behavior and provides for autonomous rover operations [Estlin *et al.*, 1999b]. In particular, this architecture provides a framework for autonomously generating and achieving planetary science goals [Estlin *et al.*, 1999a].

The overall execution architecture is shown in Figure 1. The system is comprised of the following major components:

- **Planning**: A dynamic, distributed planning system that produces rover-operation plans to achieve input rover-science goals. Planning is divided between a central planner, which efficiently divides up science goals between rovers, and a distributed set of planners which plan for operations on an individual rover. Each individual rover planner provides execution monitoring and re-planning capabilities where plans are updated as necessary in reaction to unforeseen events.
- **Data Analysis**: A distributed machine learning system which performs unsupervised clustering to model the distribution of rock types observed by the rovers. This distribution is used for prioritizing new targets for exploration by the rovers. This system is designed to direct rover sensing to continually improve the model of the scientific content of the planetary scene.

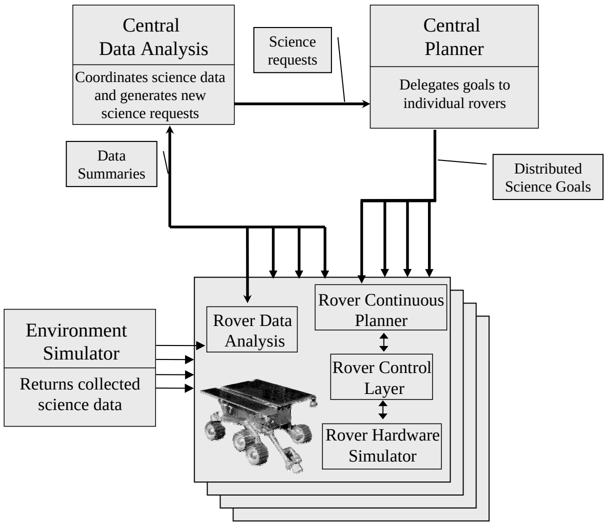

Figure 1: Multi-rover Execution Architecture

- **Rover Control Software**: Control software from the NASA JPL Rocky 7 rover that handles execution of low-level rover commands in the areas of navigation, vision and manipulation [Volpe *et al.*, 1997]. This software performs low-level monitoring and control of the rover's sub-systems.
- **Rover Hardware Simulator**: A multi-rover simulation environment that is used to simulate the rover terrain and rover hardware operations within that environment. The simulator models rover kinematics and generates sensor feedback which is relayed back to the continuous planner for each rover.
- **Environment simulator**: A multiple rover simulator that models different geological environments and rover science activities within them. The simulator manages science data for each environment, tracks rover operations within the terrain, and reflects readings by rover science instruments.

The overall system operates in a closed-loop fashion where the data analysis system can be seen to take the role of the scientist driving the exploration process. Spectra data are received by the on-board data analysis algorithms which broadcasts information to the central analysis module. This module forms a global model of the data and generates a new set of observation goals that will further improve the accuracy of the model. These goals are passed to a central planner which assigns them to individual rovers in a fashion that will most efficiently serve the requests. Then each rover planner produces a set of actions for that rover which will achieve as many of its assigned goals as possible. These action sequences are executed using the rover low-level control software and a multi-rover simulation environment which relay action and state updates to each onboard planner. As previously described, each onboard rover planner can perform re-planning when unexpected events or failures occur. Action sequences are also executed within the environment simulator and any gathered data is sent back to the rover data analysis modules. This cycle continues until enough data is gathered to produce distinct models for any observed rock types.

This architecture is currently being evaluated using the geological scenario previously described. The rest of this paper focuses on the planning aspects of this architecture. For more information on other components, please see [Estlin *et al.*, 1999a; 1999b].

4 Integrating Planning and Execution for Multi-Rover Operations

To produce individual rover plans for a team of rovers, we have developed a distributed planning environment utilizing the CASPER continuous planning system [Chien *et al.*, 1999]. CASPER is an extended version of the ASPEN (Automated Scheduling and Planning Environment) system [Fukanaga *et al.*, 1997], which has been developed to address dynamic planning and scheduling applications. CASPER provides a generic planning/scheduling application framework that can be tailored to specific domains. Its components include:

- An expressive modeling language to allow the user to naturally define the application domain
- A constraint management system for representing and maintaining domain operability and resource constraints, as well as activity requirements
- A set of search strategies and repair heuristics
- A temporal reasoning system for expressing and maintaining temporal constraints
- A graphical interface for visualizing plans/schedules
- A real-time system which monitors plan execution and modifies the current plan based on activity, state and resource updates

CASPER employs techniques from planning and scheduling to automatically generate the necessary rover-activity sequence to achieve the input goals. This sequence is produced by utilizing an iterative repair algorithm [Minton and Johnston, 1988; Zweben *et al.*, 1994] which classifies conflicts and attacks them each individually. Conflicts occur when a plan constraint has been violated where this constraint could be temporal or involve a resource, state or activity parameter. Conflicts are resolved by performing one or more schedule modifications such as moving, adding, or deleting activities. A rover that is at the incorrect location for a scheduled science activity is one type of conflict. Resolving this particular conflict involves adding a traverse command to send the rover to the designated site. Other conflicts may include having more than one rover communicating with the lander at the same time or having too many activities scheduled for one rover, which over-subscribe its power resources. The iterative repair algorithm continues until no conflicts remain in the schedule, or a timeout

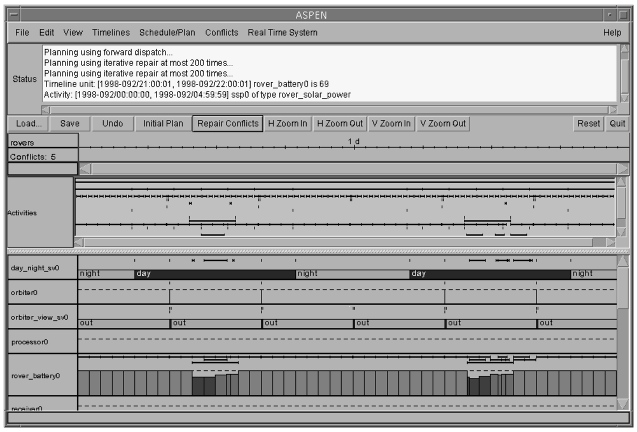

Figure 2: Example rover plan

has expired. Figure 2 shows an example plan in this domain displayed in the CASPER GUI.

4.1 Distributed Planning

To support missions with multiple rovers, we developed a distributed planning environment where it is assumed each rover has an on-board planner. This allows rovers to plan for themselves and/or for other rovers. If there is a slow communication link between rovers, or between a rover and the lander, it may be useful to have rovers construct their own plans and to re-plan dynamically when necessary. Also, by balancing the workload, distributed planning can be helpful when individual computing resources are limited.

The approach to distributed planning utilized in this environment is to include a CASPER continuous planner for each rover, in addition to a central, batch planner. The central planner develops an abstract plan for all rovers, while each agent planner develops a detailed, executable plan for its own activities. The central planner also acts as a router, taking a global set of goals and dividing it up among the agents. For example, a science goal may request an image of a particular rock without concern for which rover acquires the image. The central planner could assign this goal to the rover that is closest to the rock in order to minimize the traversals of all rovers. This master/slave design is just one approach to distributed planning; we are also experimenting with several other forms of distributed planning [Rabideau *et al.*, 1999b].

4.2 Continuous Planning for each Rover

To achieve a high level of responsiveness for each onboard rover planner, we utilize a continuous planning approach. Rather than considering planning a batch process in which a planner is presented with goals and an initial state, each rover planner has a current goal set, a current state, a current plan, and state projections into the future for that plan. At any time an incremental update to the goals or current state may update the current plan. This update may be an unexpected event or simply time progressing forward. Each planner is then

Initialize P to the null plan
Initialize G to the null set
Initialize S to the current state

Given a current plan P and a current goal set G

1. Update G to reflect new goals or goals that are no longer needed
2. Update S to the revised current state
3. Compute conflicts on (P,G,S)
4. Apply conflict resolution planning methods to P (within resource bounds)
5. Release relevant near-term activities in P to RTS for execution
6. Goto 1

Figure 3: CASPER continuous planning algorithm

responsible for maintaining a plan consistent with the most current information. The current plan is the planner's estimation as to what it expects to happen in the world if things go as expected. However, since things rarely go exactly as expected, the planner stands ready to continually modify the plan. Iterative repair techniques, as mentioned above, enable incremental changes to the goals, initial state or plan and then iteratively resolve any conflicts that may arise.

The CASPER continuous planning algorithm is shown in Figure 3. In this approach, the rover state is modeled by a set of plan timelines, which represent the current and expected state of the rover over time. At each iteration through the loop shown in the figure, the actual state of the rover drifts from the state expected by the timelines, reflecting changes in the world. As updates are relayed back from sensors and the rover control software, CASPER updates the timelines models with actual state values, resource values, start times and completion times for activities. Each of these updates, when synchronized with the current plan, may introduce conflicts (Step 3). As explained previously, a conflict is when an action in the plan is inappropriate because its required state and/or resource values violate the plan constraints.

Whenever such a conflict exists, CASPER notes the conflict and performs plan modifications to bring the plan back into sync with the current state and future-plan projections. Because this process is continuous, the plan rarely has the chance to get significantly out of sync. As a result the high-level actions of the system are more responsive to the actual rover state.

4.3 Plan Optimization

One of the dominating characteristics of the multi-rover application is rover traversals to designated waypoints. Decisions must be made not only to satisfy the requested goals, but also to provide more optimal (i.e., efficient) schedules. Both the central planner and the rover continuous planners can consider optimization goals during the repair process. As certain types of conflicts are resolved, heuristics are used to guide the search towards making decisions that will produce higher quality schedules. In other words, when several options are available for repairing a conflict, these options are ordered based on predictions on how favorable the resulting schedule will be.

For this application, we have implemented heuristics based on techniques from the Multiple-Traveling Salesmen Problem (MTSP), which is an extension of the Traveling Salesman Problem (TSP) [Johnson and McGeoch, 1997]. For MTSP, at least one member of a sales team must visit each city such that total traveling time is minimized. Both the central and individual rover planners utilize the MTSP heuristics. These heuristics are used both to select which rover should be assigned a particular science activity and to select a temporal location for the science activity. In previously reported results, they were shown to make a significant impact in reducing overall traversal distance and expected execution time [Rabideau *et al.*, 1999a].

5 Related Work

While there has been a significant amount of work on cooperating robots, most of it focuses on behavioral approaches that do not explicitly reason about assigning goals and planning courses of action. One exception is GRAMMPS [Bummit and Stentz, 1988], which coordinates multiple mobile robots visiting locations in cluttered, partially known environments. GRAMMPS also has a low-level planner on each robot and uses a similar approach to distribute targets, however GRAMMPS uses simulated annealing where we use a greedy approach for this task. Also, GRAMMPS does not look at multiple resources or exogenous events.

Many cooperative robotic systems utilize reactive planning techniques [Mataric, 1995; Parker, 1999]. These systems have been shown to exhibit low-level cooperative behavior in both known and "noisy" environments. However, these systems have not been shown useful for mission planning where a set of high-level science and engineering goals must be achieved in an efficient manner.

6 Future Work

We have several planned extensions to this work. First, we would like to extend the central (master) planner for this architecture to also utilize continuous planning techniques. Currently only the individual rover planners can perform re-planning. However, it would be beneficial to have this capability extended to the central planner which distributes science activities among the rovers. This extension would enable the central planner to reassign goals when necessary or beneficial. For instance, if a rover takes longer than expected to achieve part of its goals, it maybe useful to re-assign its remaining (unachieved) goals to other nearby rovers.

We also intend to extend the distributed planning architecture to be more robust and able to handle rover failure situations. For instance, if a rover fails the planning system should recognize this failure (e.g., the rover has not responded for a certain amount of time), refrain from sending any new goals to that rover, and re-assign any current goals assigned to that rover.

In addition, we plan to extend the planning model to represent more extensive communication between each rover. Currently, rovers share science data through the central data-analysis module. We would like rovers to also share plan information which would enable us to experiment with different forms of distributed planning, such as team-based strategies [Tambe, 1997] or market-based approaches [Smith, 1980; Sandholm, 1993] to multi-agent coordination.

Last, we plan on testing the overall execution architecture in a more realistic setting using actual rovers as opposed to the hardware and environment simulators described previously. This testing will occur in the JPL Mars yard using rovers such as JPL's Rocky 7 and Rocky 8 [Volpe *et al.*, 1997].

7 Conclusions

In this paper we have presented a distributed planning environment for coordinating multiple-rover activities. This environment utilizes continuous planning techniques to monitor plan execution for each rover and perform re-planning when necessary. These dynamic planning and re-planning techniques enable a team of rovers or act autonomously and be more responsive to unexpected changes in the environment. This system is part of a multi-rover execution architecture which is currently being tested on its ability to autonomously classify a set of terrain rocks in a Mars-like environment.

Acknowledgments

This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

References

[Bummit and Stentz, 1988] B. Bummit and A. Stentz. GRAMMPS: A generalized mission planner for multiple mobile robots in unstructured environments. In *Proceedings of the IEEE Conference on Robots and Automation*, 1988.

[Chien *et al.*, 1999] Steve Chien, Russell Knight, Andre Stechert, Rob Sherwood, and Gregg Rabideau. Integrated planning and execution for autonomous spacecraft. In *Proceedings of the 1999 IEEE Aerospace Conference*, Aspen, CO, March 1999.

[Estlin *et al.*, 1999a] Tara Estlin, Alexander Gray, Tobias Mann, Gregg Rabideau, Rebecca Castano, Steve Chien, and Eric Mjolsness. An integrated system for multi-rover scientific exploration. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, Orlando, FL, July 1999.

[Estlin *et al.*, 1999b] Tara Estlin, Jeng Yen, et al. An integrated architecture for cooperating rovers. In *Proceedings of the 1999 International Symposium on Artficial Intelligence, Robotics and Automation for Space*, Noordwijk,The Netherlands, June 1999.

[Fukanaga *et al.*, 1997] Alex Fukanaga, Gregg Rabideau, Steve Chien, and David Yan. Towards an application framework for automated planning and scheduling. In *Proceedings of the 1997 International Symposium on Artficial Intelligence, Robotics and Automation for Space*, Tokyo, Japan, July 1997.

[Johnson and McGeoch, 1997] D. Johnson and L. McGeoch. The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 215–310. John Wiley and Sons, London, 1997.

[JPL, 1999] *http://www.jpl.nasa.gov/missions/*, 1999.

[Mataric, 1995] Maja Mataric. Designing and understanding adaptive group behavior. *Adaptive Behavior*, 4(1):51–80, December 1995.

[Minton and Johnston, 1988] S. Minton and M. Johnston. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1988.

[Mishkin *et al.*, 1998] Andrew Mishkin, Jack Morrison, Tam Nguyen, Henry Stone, Brian Cooper, and Brian Wilcox. Experiences with operations and autonomy of the mars pathfinder rover. In *Proceedings of the 1998 IEEE Aerospace Conference*, Aspen, CO, March 1998.

[Parker, 1999] Lynne E. Parker. Cooperative robotics for multi-target observation. *Intelligent Automation and Soft Computing*, 5(1):5–19, 1999.

[Rabideau *et al.*, 1999a] Gregg Rabideau, Tara Estlin, and Steve Chien. Working together: Automatic generation of command sequences for multiple cooperating rovers. In *Proceedings of the 1999 IEEE Aerospace Conference*, Aspen, CO, March 1999.

[Rabideau *et al.*, 1999b] Gregg Rabideau, Tara Estlin, Steve Chien, and Tony Barrett. A comparison of coordinated planning methods for cooperating rovers. In *Submitted to the AIAA 99 Space Technology Conference*, Albuquerque, NM, September 1999.

[Sandholm, 1993] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washinton, D.C., July 1993.

[Smith, 1980] G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12), 1980.

[Tambe, 1997] Milind Tambe. Towards flexible teamwork. *Journal of Artificial Intelligence Research*, 7:83–124, 1997.

[Volpe *et al.*, 1997] R. Volpe, J. Balaram, T. Ohm, and R. Ivlev. Rocky 7: A next generation mars rover prototype. *Journal of Advanced Robotics*, 11(4), 1997.

[Zweben *et al.*, 1994] M. Zweben, B. Daun, E. Davis, and M. Deale. Scheduling and rescheduling with iterative repair. In Monte Zweben and Mark Fox, editors, *Intelligent Scheduling*, pages 241–256. Morgan Kaufmann, San Francisco, CA, 1994.