AN INTEGRATED ARCHITECTURE FOR COOPERATING ROVERS

T. Estlin, J. Yen, R. Petras, D. Mutz, R. Castaño, G. Rabideau,
R. Steele, A. Jain, S. Chien, E. Mjolsness, A. Gray, T. Mann, S. Hayati, H. Das

Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive, M/S 126-347, Pasadena, CA 91109-8099
phone: +1 818 393-5375, fax: +1 818 393-5244, email: {firstname.lastname}@jpl.nasa.gov

ABSTRACT

This paper presents a rover execution architecture for controlling multiple, cooperating rovers. The overall goal of this architecture is to coordinate multiple rovers in performing complex tasks for planetary science. This architecture integrates a number of systems and research efforts on single rovers and extends them for multiple rover operations. Techniques from a number of different fields are utilized, including AI planning and scheduling, real-time systems and simulation, terrain modeling, and AI machine learning. In this paper, we discuss each architecture component, describe how components interact and present the geological scenario we are using to evaluate the overall architecture.

1. INTRODUCTION

This paper describes an integrated architecture being developed at the NASA Jet Propulsion Laboratory for solving planetary surface exploration problems though the utilization of multiple cooperating rovers. Utilizing multiple rovers for science and exploration activities has a number of advantages. First, we can greatly increase mission science return by simultaneously using complementary instruments on different rovers and efficiently dividing science-gathering tasks between the rovers. Second, multiple rovers can perform tasks that otherwise would not be possible using a single rover, such as taking wide baseline stereo images. Third, multiple rovers would 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.

This paper presents work in demonstrating how multiple rovers as compared to a single rover can more effectively explore a selected site and return more science data per communication cycle. The described architecture utilizes research results on single rovers (i.e. command sequence generation, navigation, control, science operations, ground control, etc.) and extends them to multiple rovers. An integrated system architecture has been developed that can automatically generate interesting science goals, plan for and coordinate multiple rover activities, and monitor and update activities in response to anomalous events. This architecture also utilizes a multi-rover simulation environment and control software from the NASA JPL Rocky 7 rover [Volpe et al., 1997]. Techniques from several different fields are combined including Artificial Intelligence (AI) planning and scheduling, real-time systems and simulation, terrain modeling and system kinematics/ dynamics, and AI machine learning.

The organization of this architecture consists of the following. An AI planning and scheduling system (CASPER) takes as input a set of science goals for exploring a particular terrain and then automatically generates plans (i.e. command sequences) that coordinate a team of rovers in successfully completing the goals and exploring the requested areas. Each rover plan is then relayed to the onboard control software and executed in a multi-rover simulation environment (ROAMS) that is used to simulate the rover terrain and rover operations within that environment. The simulator also generates sensor feedback from the rovers which is relayed back to the planner. This feedback is utilized to determine the success or failure of certain activities and any changes in resources or states. If unexpected changes have occurred, the planning system can perform re-planning to fix the original plan and ensure the successful achievement of the goals. An AI clustering algorithm analyzes any science data gathered by the rovers and then uses this analysis to produce new science goals for the rovers to accomplish. This architecture is currently being evaluated using a geological scenario where rovers are used to examine and classify terrain rocks.

The remainder of this paper is organized as follows. We begin by characterizing the multiple cooperating rovers application domain and describing the particular science scenario we are using to evaluate our integrated system. Next, we present out multi-rover execution architecture which controls and coordinates operations for a team of rovers. We then describe each of its individual components and any interactions between them. In the final sections, we discuss related work, planned future work, and present our conclusions.

2. COOPERATING ROVERS FOR SCIENCE

Utilizing multiple rovers on planetary science missions has several important advantages:

- *Force multiplication*. Multiple rovers can collect more data than a single rover and can perform certain types of tasks more quickly than a single rover, such as: performing a geological survey of a region or deploying a network of seismographic instruments. We call these *cooperative* tasks.
- *Simultaneous presence*. Multiple rovers can perform tasks that are impossible for a single rover. We call these *coordinated* tasks. Certain types of instruments, such as interferometers, require simultaneous presence at different locations. Rovers landed at different locations can cover areas with impassable boundaries. Using communication relays, a line of rovers can reach longer distances without loss of contact. More complicated coordinated tasks can also be accomplished, such as those involved in hardware construction or repair.
- *System redundancy*. Multiple rovers can be used to enhance mission success through increased system redundancy. Several rovers with the same capability may have higher acceptable risk levels, allowing one rover, for example, to venture farther despite the possibility of not returning. Also, because designing a single rover to survive a harsh environment for long periods of time can be difficult, using multiple rovers may enable missions that a single rover could not survive long enough to accomplish.

In all cases, the rovers can behave in a cooperative or even coordinated fashion, accepting goals for the team, performing group tasks and sharing acquired information.

Coordinating distributed rovers for a mission to Mars or other planet introduces some interesting new challenges for the supporting technology. Issues arise concerning interfaces, communication, control and individual on-board capabilities. For example, different software components must successfully interface onboard the rovers to provide the needed autonomous functionality. In addition, mission designers will need to decide on interfaces among the rovers, to the lander and/or orbiter and to the ground operations teams. Decisions will need to be made on communication capabilities, which will limit the amount of information shared between rovers and the lander/orbiter. A distributed control protocol will need to be selected that defines how tasks are distributed among rovers and the “chain of command” for the rovers. Finally, the onboard capabilities will need to be considered, including computing power and onboard data-storage capacity.

Many of these design decisions are related, and all of them have an impact on the onboard technologies that can be utilized by the mission. The interfaces determine what activities can be planned for each rover and what data or sensor feedback can be utilized by the onboard software. The amount of communication available will determine how much science or terrain data can be shared among rovers and will affect how much each rover can coordinate with other rovers to perform tasks. In addition, communication capabilities will affect the amount of onboard autonomy required. If bandwidth is low and reaction time is critical, a rover will need to react intelligently to the environment, including performing autonomous navigation and replanning for its own activities in response to unexpected events. The control scheme will determine which rover executes which activities and which rovers coordinate and monitor activities of the others. Decisions on the onboard capabilities of each rover limit the independence of the rover. With little computing power, a rover may only be able to execute commands. More power may allow it to plan command sequences, replan if necessary, and analyze gathered data. Some rovers may also perform these activities as a service to other rovers or in cooperation with them.

To evaluate the architecture presented in this paper, we have initially chosen the configuration of a team of three rovers where each rover has a planning and data-analysis tool onboard as well as low-level control software for tasks such as navigation and vision. Each rover can thus plan for its assigned goals, execute and monitor generated commands, collect the required data, perform re-planning if necessary, and perform science analysis onboard to direct its future goals.

Currently we are evaluating our framework by testing its ability to build a model of the distribution of surrounding terrain rocks, classified according to composition as measured by a boresighted spectrometer. Science goals consist of requests to take spectral measurements at certain locations or regions. These goals are prioritized so that, if necessary, low priority goals can be preempted (e.g., due to low battery power). Science goals are divided among the three rovers. Each rover is identical and is assumed to have a spectrometer onboard as well as other resources including 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 can be transferred to an orbiter where it is stored in memory.

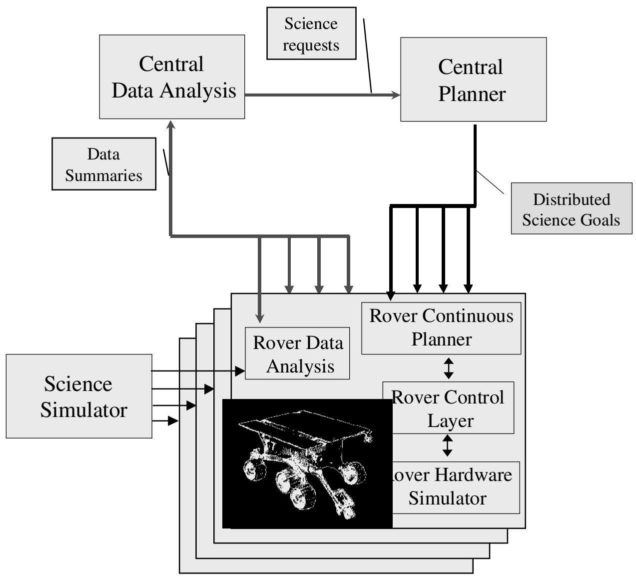

Figure 1: Multi-rover Execution Architecture

3. MULTI-ROVER EXECUTION ARCHITECTURE

The overall system 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 among rovers, and a distributed set of planners, which plan for operations on individual rovers and can perform re-planning if necessary.
- **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. This software performs low-level monitoring and control of each rover's subsystems.
- **Multi-Rover Real-Time Simulator:** A multi-rover simulation environment that is used to simulate the planetary terrain and rover hardware operations within that environment. This simulator models rover kinematics and generates sensor feedback which is relayed back to each rover planner.
- **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 also used for prioritizing new targets for exploration by the rovers.
- **Science Simulator:** A multi-rover science simulator that models different geological environments and rover science activities within them. The science simulator manages science data for the current terrain, tracks rover operations within that terrain, and reflects readings by rover science instruments.

The overall system operates in a closed-loop fashion. Science goals (e.g., a spectrometer reading at a certain location) are given to a central planner which assigns them to individual rovers in a fashion that will most efficiently serve the requests. Each rover planner then 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 hardware simulation environment which relay action and state updates back to each onboard planner. If necessary, each onboard planner can perform re-planning when unexpected events or failures occur.

Action sequences are also executed within the science simulator and any gathered data is sent to the rover data-analysis modules. These modules form local models of the observed data that are broadcast to the central analysis module. This module forms a global rock-distribution model and generates a new set of observations goals that will further improve the accuracy of the model. In this way, the data analysis system can be seen to take the role of the scientist driving the exploration process. New science goals are then sent to the centralized planner and the overall cycle continues until enough data is gathered to produce distinct models for any observed rock types.

In the next few sections, we discuss each of the architecture components in more detail.

3.1 DISTRIBUTED, CONTINUOUS PLANNING

To produce individual rover plans for a team of rovers, we have developed a distributed planning environment utilizing the CASPER planning system [Chien et al., 1999]. CASPER (Continuous Activity Scheduling, Planning, Execution and Replanning) is an extended version of the ASPEN system [Fukanaga et al., 1997] that has been developed to address dynamic planning and scheduling applications. CASPER employs techniques from AI 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

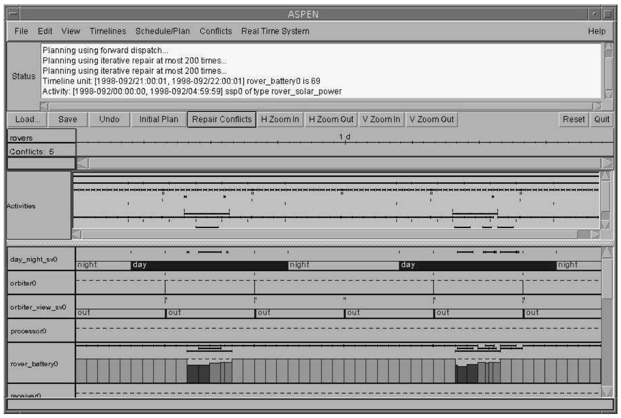

Figure 2: Example Rover Plan

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 an activity. Examples of conflicts include a rover that is at the incorrect location for a scheduled science activity or having too many activities scheduled for one rover, which oversubscribes its power resources. Figure 2 shows an example rover-plan displayed in the CASPER GUI.

To support missions with multiple rovers, we developed a distributed planning environment where it is assumed each rover has an onboard planner. This allows rovers to plan for themselves and/or for other rovers. And, by balancing the workload, distributed planning can be helpful when individual computing resources are limited. Our approach to this task was 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 science goals and dividing it up among the separate rovers. 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 approach is just one approach to distributed planning which could be utilized for this architecture; we are also experimenting with several other forms of distributed planning for this task [Rabideau, et al., 1999].

In order to enhance the quality of the produced schedules, we have implemented heuristics for assigning rovers to goals and for deciding on the order in which to visit each of the specified locations. The heuristics borrow from algorithms for finding solutions to the Multiple Traveling Salesman Problem (MTSP) [Johnson et al., 1997]. With multiple rovers covering the same area, the planner prefers paths that minimize the total traverse time of all the rovers.

To achieve a high level of responsiveness for each onboard rover planner, we also 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 onboard planner is then responsible for maintaining a plan consistent with the most current information obtained from the rover sensors and low-level control software. 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 to bring it back into sync with the actual rover state.

3.2 ROVER CONTROL SOFTWARE

To handle low-level rover control issues, we utilize the Onboard Rover Control & Autonomy Architecture (ORCAA) software developed for the NASA JPL Rocky 7 rover [Volpe et al., 1997, Hayati & Arvidson, 1997]. In the ORCAA software, asynchronous rover activities are initiated by a queue of rover commands. These activities are represented using asynchronous finite state machines (FSMs) and synchronous data-flow control loops. When the rover receives a command sequence, these commands cause state transitions in one of three main state machines: Navigation, Vision and Manipulation. For example, in the Navigation FSM, possible states include "Idling", "Steering", "Driving", etc. State transitions in these FSMs are used to run different execution methods and are often used to begin the execution of synchronous processes, which perform monitoring and control of the rover's subsystems.

This software also relays sensor information and command updates back to the overlying planning system. This information includes command updates such as whether a command was successfully executed and sensor values such as the current sun angle or level of battery power.

3.3 MULTI-ROVER REAL-TIME SIMULATION

In order to accomplish preliminary testing of this architecture, a real-time simulation environment has been developed using the DARTS/Dshell software [Biesiadecki, et al., 1997]. The Rover Analysis Modeling and Simulation (ROAMS) [Yen et al., 1999] extension of DARTS/Dshell was first slated towards modeling single-rover operations and is based on the Rocky 7 Mars rover. Currently, the simulator rover model is comprised of its mechanical, electrical, and sensor subsystems, and is connected with the on-board (Rocky 7) software. Several terrain models have been incorporated and we have developed solution techniques that permit a real-time simulation of the rover traversing a Mars-like terrain on a workstation platform.

The basic component of the simulator is the solution of inverse kinematics for the rover traversing a Mars-like terrain. Building on this novel solution technique, we have applied the ROAMS rover simulator to testing the Rocky 7 on-board software. The control and navigation algorithms of the control software are used to drive the Rocky 7 rover model against a terrain with randomly distributed rocks. Applying the DARTS/Dshell methodology, we implemented models for hardware devices, such as a panoramic spectrometer, sun sensor, tilt sensor, obstacle detection camera, solar panel, battery, etc., to feed the subsystems. Also, based on the numerical solution of inverse kinematics, the hardware instrument models provide high-fidelity synthetic data to test the control and navigation code. Overall, this environment permits a fast and better design and implementation of the rover's software subsystem.

For the multiple rover architecture, this single-rover simulation model has been extended to support several cooperating rovers. An example situation involving three rovers is shown in the ROAMS interface in Figure 3. For use with this architecture, we developed additional hardware models, including a collision avoidance model, an obstacle detection model, models of power units, and the capability for running multiple rovers in ROAMS. Due to the stability and accuracy of the numerical solution, these device models can provide high quality sample data for the control software and ultimately the planning system. For example, the power source of Rocky 7, including a solar panel and a battery, can produce accurate reading of the power level due to the prediction of the panel's attitude and the wheel's motor output. As explained above, these and other sensor values

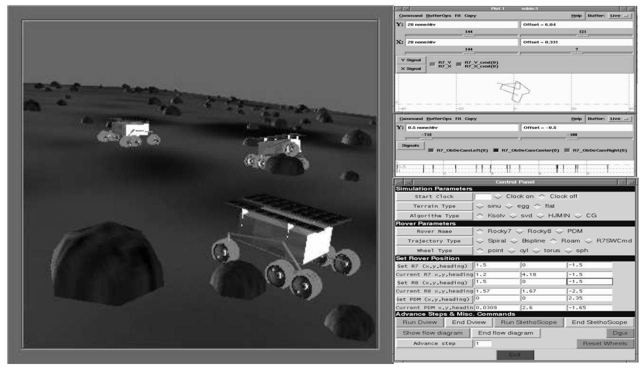

Figure 3: Three rovers in sample terrain

can be fed back to each rover planner so that a valid command sequence can be consistently maintained for each rover.

3.5 SCIENCE-DATA ANALYSIS

To perform science analysis, we use a machine-learning system which performs unsupervised clustering to model the distribution of rock types in the observed terrain [Estlin, et al., 1999]. Clustering is performed by a distributed algorithm where each rover alternates between independently performing learning computations using its local data and updating a global-distribution model through communication among rovers. The model used for this distribution is a simple *K*-means-like unsupervised clustering model, where each cluster represent a different rock type in the sensor space. Currently, each sensor reading is a spectral measurement returning values at 14 wavelengths; learning takes place in the full 14-dimensional continuous space. A sample cluster model (shown for 2 of 14 dimensions) is shown in Figure 4.

After a new set of science readings is acquired, each rover sends a small set of parameters, which summarizes its local data, to the central clusterer. The central module then integrates this data into an updated global model and broadcasts that model to all rovers in the system. This process continues iteratively until convergence.

Output clusters are also used to help evaluate visible surfaces for further observation based upon their "scientific interest." Specifically, the system tries to increase the accuracy of the clustering model by obtaining data readings in regions that are likely to improve the model. Each update of the global clustering model determines a new set of interesting science goals, i.e. planetary locations to be explored by the rovers.

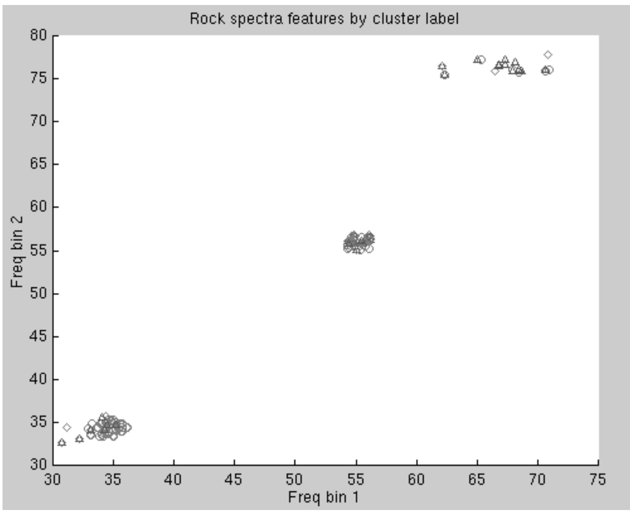

Figure 4: Example spectra-feature space

These observation goals are then sent as formal goals by the learner to the planner. Thus, the science analysis system can be viewed as driving the science process by analyzing the current data set and then deciding what new and interesting observations should be made.

3.5 SCIENCE SIMULATOR

A science simulator designed for this particular geological scenario provides data for the science-analysis system by simulating the data-gathering activities of the rover. Different Martian rockscapes are created for use in the simulator by using distributions over rock types, sizes and locations. The size and spatial distributions of the rockfield were developed by examining distributions of rocks observed by the Viking Landers, Mars Lander and Mark Pathfinder. The distribution of minerals that can occur in rocks was developed in collaboration with planetary geologists at JPL, and the spectra associated with rocks are generated from the spectra of the component minerals via a linear-mixing model. When science measurements are requested from a terrain during execution, rock and mineral spectral models are used to generate sample spectra based on the type of rock being observed. This data is then communicated to the relevant rover data-analysis module.

4. 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 [Bummitt 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, however it does not look at multiple resources or exogenous events. It also does not utilize a learning system to analyze gathered data and deduce new goals.

Many cooperative robot 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 high-level set of science and engineering goals must be achieved in an efficient manner.

The idea of having a scientific-discovery system direct future experiments is present is a number of other systems [Rajamoney, 1990; Nordhausen and Langley, 1993], however none of these have been utilized for multiple-robot scenarios. In our architecture, the data-analysis system is integrated with a planning system and real-time simulator, which plan and execute detailed activity sequences needed to perform each experiment. The data-analysis system also directly interacts with the environment and is specialized to problems and scenarios in planetary science.

5. FUTURE WORK

We have a number of planned extensions to this work. First, we intend to extend the overall architecture to be more robust and able to handle rover failure situations. For instance, if a rover fails, the distributed 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.

Another important addition is to integrate the Envelope Learning and Monitoring using Error Relaxation (ELMER) system [Decoste, 1997] to model rover-resource use such as battery power or onboard memory. ELMER uses statistical machine-learning techniques to learn and refine input-conditional limit functions from historic and/or simulated data. These limit functions define context-sensitive upper and lower boundaries, within which future resource-data is expected to fall. This system will enable more accurate resource modeling, which can be used by the planner to better estimate future resource levels.

We also plan to increase the fidelity of the simulation by adding models of onboard cameras and other instruments, and extending the simulator to model communication between each rover. Currently, it is assumed rovers share science data through the central data-analysis model, however this communication is not explicitly represented in the simulator. We would also like rovers to share plan information, which would allow them to directly coordinate with each other during plan execution and would allow us to experiment with different forms of distributed planning that require communication among agents [Tambe, 1997; Sandholm, 1993].

Last, we plan on testing the overall architecture in a more realistic setting using actual rovers as opposed to the hardware and science simulators described previously. This testing will occur in the JPL Mars yard and/or in outside field tests using rovers such as JPL's Rocky 7 and Rocky 8.

6. CONCLUSION

In conclusion, using multiple rovers can greatly increase the capabilities and science return of a mission. In this paper we have presented an integrated architecture that combines techniques from several fields to effectively plan for and coordinate rover activities, execute these activities in a real-time environment simulator, monitor rover-execution status, and effectively respond to unexpected events through re-planning. This integrated system exhibits great potential for advanced applications in areas of design, engineering, and distributed planning for mobile robotic systems.

ACKNOWLEDGEMENTS

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

REFERENCES

Biesiadecki, J., Henriquez, D., and Jain, A., "A Reusable, Real-Time Spacecraft Dynamics Simulator," *Proceedings of the Sixth Digital Avionics Systems Conference*, Irvine, CA, October 1997.

Bummit, B., and Stentz, A., "GRAMMPS: A Generalized mission planner for multiple robots in unstructured environments," *Proceedings of the IEEE Conference on Robotics and Automation*, 1988.

Chien, S., Knight, R., Stechert, A., Sherwood, R., and Rabideau, G., "Integrated Planning and Execution for Autonomous Spacecraft," *Proceedings of the 1999 IEEE Aerospace Conference*, Aspen, CO, March, 1999.

Estlin, T., Gray, A., Mann, T., Rabideau, G., Castano, R., Chien, S., and Mjolsness, E, "An Integrated System for Multi-Rover Scientific Exploration," *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (AAAI-99), Orlando, FL July 1999.

Decoste, D., "Mining Multivariate Time-Series Sensor Data to Discover Behavior Envelopes," *Proceedings of the Third Conference on Knowledge Discovery and Data Mining* (KDD-97), Newport Beach, CA, August 1997.

Fukanaga, A., Rabideau, G., Chien, S., and Yan, D., "Toward an Application Framework for Automated Planning and Scheduling," *Proceedings of the 1997 International Symposium of Artificial Intelligence, Robotics and Automation for Space* (iSAIRAS-97), Tokyo, Japan, July 1997

Hayati, S., and Arvidson, R., "Long Range Science Rover (Rocky 7) Mojave Desert Field Tests," *Proceedings of the 1997 International Symposium on Artificial Intelligence, Robotics and Automation in Space* (iSAIRAS-97), Tokyo, Japan, July 1997.

Johnson, D. and McGeoch, L., "The Traveling Salesman Problem: A Case Study in Local Optimization." *Local Search in Combinatorial Optimization*, Aarts, E. and Lenstra,, J., eds., John Wiley and Sons, London, 1997, pp. 215-310.

Mataric, M., "Designing and Understanding Adaptive Group Behavior," *Adaptive Behavior*, 4(1):51-80, December 1995.

Minton, S., and Johnston, M. "Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems," *Artificial Intelligence*, 58:161-205, 1988.

Nordhausen, B., and Langley, P., “An Integrated Framework for Empirical Discovery,” *Machine Learning* 12:17-47.

Parker, L., “Cooperative Robotics for Multi-Target Observation,” *Intelligent Automation and Soft Computing*, 5(1):5-19, 1999.
Rabideau, G., Estlin, T. Chien, S. and Barrett, T., “A Comparison of Coordinated Planning Methods for Cooperating Rovers,” Submitted to the *AIAA99 Space Technology Conference*, Albuquerque, NM, September, 1999.

Rajamoney, S., “A Computational Approach to Theory Revision,” *Computational Models of Scientific Discovery*, Shrager, J., and Langley, P., eds., Morgan Kaugman, 1990.

Sandholm, T., “An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations,” *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C., July 1993.

Tambe. M., “Towards Flexible Teamwork,” *Journal of Artificial Intelligence Research*, 7:83-124, 1997.

Volpe, R., Balaram, J., Ohm, T., and Ivlev, R., “Rocky 7: A Next Generation Mars Rover Prototype,” *Journal of Advanced Robotics*, 11(4), December 1997.

Yen, J., Jain, A., and Balaram, J., “ROAMS: Rover Analysis, Modeling and Simulation Software,” *Proceedings of the Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space*, Noordwijk, The Netherlands, June 1999.

Zweben, M., Daun, B., Davis, E., and Deale, M., “Scheduling and Rescheduling with Iterative Repair,” *Intelligent Scheduling*, Zweben, M., and Fox, M., eds., Morgan Kaufman, 1994, pp.241-256.