A Comparison of Coordinated Planning Methods for Cooperating Rovers

Gregg Rabideau, Tara Estlin, Steve Chien, Anthony Barrett

Jet Propulsion Laboratory,
California Institute of Technology
4800 Oak Grove Drive, M/S 126-347
Pasadena, CA 91109
{firstname.lastname}@jpl.nasa.gov

Abstract

This paper describes and evaluates three methods for coordinating multiple agents. These agents interact in two ways. First, they are able to work together to achieve a common pool of goals which would require greater time to achieve by any one of the agents operating independently. Second, the agents share resources that are required by the actions needed to accomplish the goals. The first coordination method described is a centralized scheme in which all of the coordination is done at a central location and the agents have no autonomy at the planning level. The second method performs goal allocation using a centralized heuristic planner and (distributed) planners for the individual agents perform detailed planning. The third method uses a contract net protocol to allocate goals and then (distributed) planners for the individual agents perform detailed planning. We compare these approaches and empirically evaluate them using a geological science scenario in which multiple rovers are used to sample spectra of rocks on Mars.

1. Introduction

Significant events have recently taken place in the areas of space exploration by planetary rovers. The Mars Pathfinder and Sojourner missions were major successes, not only demonstrating the feasibility of sending rovers to other planets, but also demonstrate the utility of such missions to the scientific community. Further missions are being planned to send robotic vehicles to Mars (Mars01, Mars03, Mars05), an asteroid (MUSES-CN) (JPL 1999) as well as the outer planets and their moons. In order to increase science return and enable new types of observations new missions are being proposed that employ larger sets of robotic workers. Whether it is an unmanned spacecraft with remote rovers, or a mix of humans and robotic assistants, the command and control task for these machines will be complex. While manual request and sequence generation was possible for Sojourner, new missions will need to automate much of this process.

While it is up to mission designers to determine the optimal number of rovers for a given mission, multiple rovers have three types of advantages over single rover approaches: force multiplication, simultaneous presence and system redundancy.

- *Force multiplication*. Multiple rovers 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 a 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.

Whether they are spacecraft, probes or rovers, coordinating multiple distributed agents introduces unique challenges for automated planning and other supporting technology (Mataric 1995; Parker 1998). Issues arise concerning interfaces between agents, communication bandwidth, group command and control, and onboard capabilities. For example, a certain level of communication capabilities will need to be assigned to each, possibly limiting the amount of information that can be shared between the rovers (and ground). The mission design will need to include a "chain of command" for the team of spacecraft/rovers, indicating which rovers are controlled directly from the ground, and which are controlled by other rovers or orbiting/landed spacecraft. Finally, the onboard capabilities will need to be

considered, including computing power and onboard data storage capacity. This will limit the level of autonomy each of the rovers can have.

Many of these design issues are related, and all of them have an impact on possible automated planning and scheduling for the mission. The interfaces determine what activities can be planned for each rover. The amount of communication available will determine how much each rover can share its plan. The control scheme will also determine which rovers execute activities in the plans. If one rover controls another, the "master" rover will send activities from its plan to the "slave" rover for execution. Decisions on the onboard capabilities of each rover, however, will limit the independence of each rover. With little computing power, one rover may be unable to plan and may only be able to execute commands. More power may allow it to plan and execute. Still more power may allow a rover to plan for itself and other rovers.

In our approach, we examine the use of Artificial Intelligence (AI) planning and scheduling in three different control structures to automatically generate appropriate low-level rover command sequences to achieve science goals. In the three approaches, we explore a range of distribution of the planning function ranging from a completely centralized planner to a bidding system in which the planning process occurs on each rover in parallel. Other approaches to multi-agent planning have various degrees of distribution (Mataric 1995; Parker 1998; Hagopian, Maxwell, and Reed 1994; Cook, Gmystrasiewicz and Holder 1996; Fischer et al. 1995; Müller 1996).

This rest of this paper is organized in the following manner. We begin by characterizing the multiple cooperating rovers application domain and describe some of the interesting challenges. Next, we introduce the ASPEN planning and scheduling system and explain how automated planning and scheduling techniques can be applied to this problem. We discuss several heuristics for solving the MTSP problem and present some results on how they improve both system and final plan efficiency. We then discuss the overall framework that is used to achieve a set of geology related science goals. Next, we discuss both how to extend this system to provide the long-term goal of rover and spacecraft autonomy and how this extension compares with related efforts. Finally, we present our conclusions and discuss several of the issues being addressed in future work.

2. Baseline Scenario

We evaluate the architectures presented in this paper using the following geological scenario. It takes three steps to produce a terrain model and a set of science goals over that model. The first step creates different Martian rockscapes 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 if necessary, low priority goals will be deleted first. Upon requesting spectral measurements from this terrain during execution, rock and mineral spectral models define how to generate sample spectra based on the type of rock being observed.

Science goals are generated from experiments using a (machine learning) clustering algorithm that evaluates the current spectral data available for a particular landscape and then determines new science goals to be achieved. Rather than sampling over the spatial distribution of rocks, the clustering algorithm generates science goals (i.e. spectral readings) that will best classify rock types.

In each architecture science goals are divided among three identical rovers. Each rover has several science instruments on board including a camera and a spectrometer. Other onboard resources include a drive motor, a solar-array panel that provides power for all rover activities, and a battery that provides backup power when no solar-array power is available. The battery can also be recharged using the solar-array when solar power is not being used to capacity. Collected science data is immediately transmitted to a 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 view.

Formulating plans in this scenario involves dividing goals between rovers in a method that minimizes the amount of driving each rover must perform. Decisions must be made not only to satisfy the requested goals, but also to provide more optimal schedules. When assigning a goal to a rover, the architecture must select the best rover for the job and decide the order that each rover will achieve its assigned goals. These decisions are further complicated by the state and resource constraints mentioned above. For instance, communication constraints between the rover and orbiter may affect when certain science operations can be performed. Low priority goals may also be deleted if a rover is unable to achieve them due to temporal or resource constraints.

2.1. ASPEN Planner

All of our architectures require a planner/scheduler to turn abstract science goals into concrete activity schedules, and we extend the ASPEN (Fukunaga et al. 1997) application framework to satisfy this requirement. Using ASPEN involved generating models of the lander and rovers in the ASPEN modeling language. This language lets us define the set of activities, resources, and state variables as well as the interactions and constraints between them. The application model essentially defines the types of activities and resources that can occur in a given schedule. Figure 1 shows some examples of activity types for the multiple rover domain. A plan/schedule is a particular configuration of instances of the activity and resource types. Some activities are uncontrollable but may have effects that are required by other activities. For example, sunrise and sunset determine when solar panels are operational. These

activities are simply loaded at the start of planning. Next, the high-level science goals can be inserted into the schedule. Typically, these are unexpanded activities that have unspecified parameter values, including the start time. In addition, goals will usually have unsatisfied requirements that can only be resolved with other activities. From this, the planner/scheduler must generate a plan that has all of these problems resolved.

```
Activity rover1_image {
  int x, y, z; // location of image
  Reservations =
    rover1_battery use 10,
    rover1_memory use 1000,
    rover1_location must_be <x,y,z>;
}

Activity rover1_goto {
  int x, y, z; // location to go to
  Reservations =
    rover1_battery use 100,
    rover1_location change_to <x,y,z>;
}
```

Figure 1. Rover Activity Definitions

In ASPEN, unexpanded activities, unspecified parameter values, unsatisfied requirements and violated constraints are all considered conflicts in the schedule. Therefore, the problem becomes one of finding a conflict-free schedule. ASPEN has a library of algorithms designed to search for a conflict-free schedule. One of the more widely used algorithms is based on a technique called "iterative repair" (Zweben et al 1994). In this algorithm, conflicts are classified and addressed in a local, iterative fashion. First, a conflict from the set of conflicts is chosen for repair. Then, a repair method is chosen as an operation for resolving the conflict. Repair methods include moving activities, adding new activities, deleting activities, setting parameter values, and decomposing activities into subactivities. For each method, other decisions may be required. For example, when moving, an activity and location must be selected. When setting a parameter, a new value must be chosen. After making the appropriate decisions, the scheduling operation is performed in hopes of resolving the conflict. Finally, the new set of conflicts is collected, and the algorithm repeats until no conflicts are found, or a user-defined resource bound has been exceeded.

2.2. MTSP Heuristics

One of the dominating characteristics of the multi-rover application is the 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. When not considering efficiency, one possible schedule that achieves all science goals is to send one rover to every target location. However, usually this would not be the desired behavior, and therefore some schedule optimization must be done. We have chosen to do this optimization during the repair process. As certain types of conflicts are resolved, heuristics are used to guide the search into making decisions that will produce more optimal 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.

The heuristics we have implemented are based on techniques from the Multi-Traveling Salesmen Problem (MTSP). The Traveling Salesman Problem (TSP) (Johnson & McGeoch 1997) is one of finding a minimal tour for a salesman that needs to visit a number of cities (and typically return home). For MTSP, at least one member of a sales team must visit each city such that total traveling time is minimized. Salesmen are allowed to travel in parallel with each other.

Many algorithms exist for solving both TSP and MTSP problems. For a small number of locations ($N<10$) optimal solutions can be found in a reasonable amount of time. However, for larger sets of locations, finding optimal solutions is too expensive (NP-hard) and approximate algorithms can be used (Hochbaum 1997). Greedy techniques can be used to find near optimal solutions in polynomial time ($O(N^2)$), where the resulting tour lengths have been proven to be at most $\lceil \lg N \rceil + 1$ times the optimal length. One such technique involves taking unvisited locations and incrementally inserting each into an existing planned tour between locations where it would cause the smallest increase in tour length. We can easily extend this algorithm to multiple travelers. Unvisited locations are inserted into *any* of the tours when looking for the shortest tour.

The multi-rover scenario fits naturally into the MTSP class of problems, with only a few differences. First, the rovers are typically not required to return to their original locations (however, for sample return missions, this would be necessary). This is a minor difference and does not change the general problem[1]. Figure 2 shows three possible insertions (one from each path) for a new location. Second, while planning activities for multiple rovers, one is also be concerned with the earliest finish time (i.e., makespan) of the schedule. The schedule with the minimum total path length (sum of rover path lengths) may not necessarily be the schedule where all activities finish the earliest. Reducing the total traverse time will reduce wear on the rovers, while reducing the makespan will increase the available science time. Finally, generating command sequences requires reasoning about more than just the paths of the rovers. Each rover has a set of flight rules and a limited amount of resources. All commands, including traverses, must be scheduled in a way that does not violate any of the flight rules or resource constraints. Some of

[1] We use the term "path" as opposed to "tour" to distinguish from traversals that return to the original location. Here, a path is a traversal between science waypoints. We do not address path planning for the purpose of obstacle avoidance.

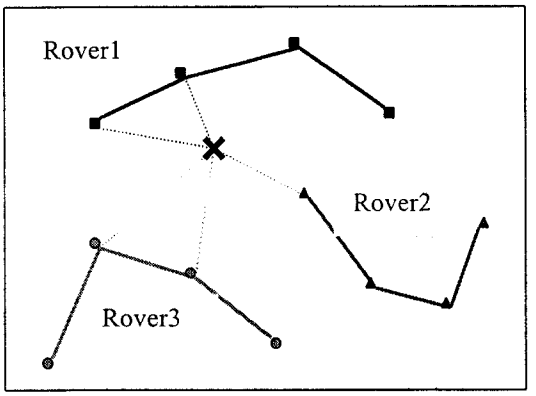

Figure 2. Traveling Rovers

these constraints may inherently require sub-optimal travel paths.

3. Many Architectures for Coordination

In the multi-rover application, activities and resources are modeled for the lander and each of the rovers. The lander provides the communication link as well as temporary data storage. Each rover has activities such as traversing, turning, taking images, taking spectrometer readings, and digging. Each rover has its own resources such as battery power, solar array power, and science instruments, and state variables representing location and orientation. If we let an ASPEN process execute on both the lander and each of the rovers, we have to decide how to distribute the model across the processes and how to coordinate them.

While there are many approaches to coordinating a set of agents, the two most common either treats them as a single master agent directing a set of slaves or treats them as a set of competing peers. Actually, these two architectures determine a whole spectrum of architectures where a master agent gives its slaves progressively more autonomy. In this section we describe the two extreme approaches and an intermediate one. In each case the collection of ASPEN processes interact to follow the heuristics characterized by the greedy insertion MTSP algorithm. While the master can run on the ground, on an orbiter, on a lander, or on one of the rovers, we simplify our presentation by always treat the lander as a master with slave rovers.

3.1. Centralized Planning

The master/slave approach to automated planning for multiple agents involves using a single centralized planner. As shown in figure 3, planning and scheduling for all agents is done with a single ASPEN process on the lander, this approach only needs one planning model to represent the collection of activities, resources and constraints associated with every agent. When planning is complete, the relevant sub-plans (i.e., command sequences) are transmitted to each "slave" rover for execution.

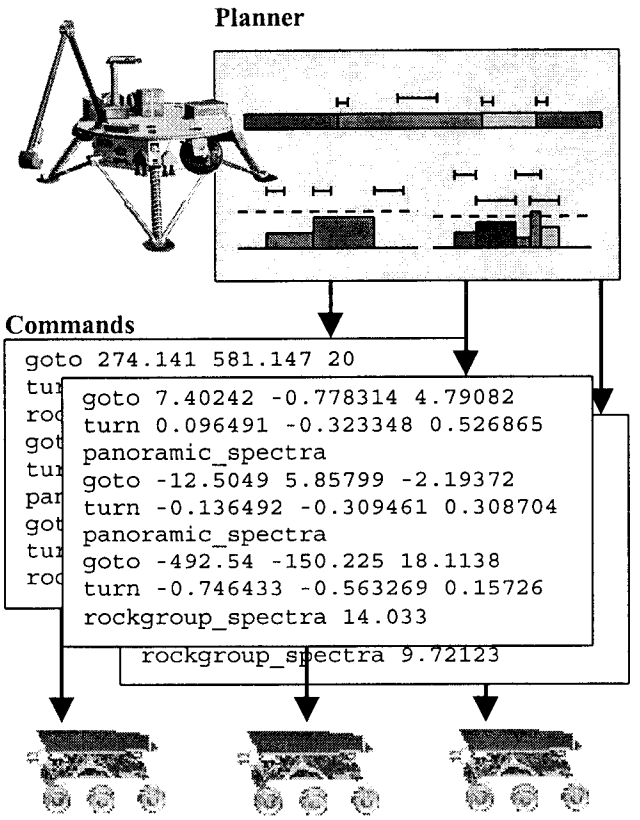

Figure 3. Centralized Planning

Implementing this approach involves collecting all of the models and adding abstract activities with decompositions to determine which rover to use for a particular science goal. For instance, there are three ways to decompose an abstract activity to take a picture – one for each of the rovers. Its decomposition determines which of the three rovers to use.

When generating command sequences for multiple rovers, ASPEN uses two heuristics that implement a greedy insertion MTSP algorithm. One is used to select a decomposition of a generic science goal into a specific science activity for one of the rovers. The other is used to select a temporal location for the science activities when they are moved. Both use the same evaluation criteria: make the selection that results in the shortest path. For the decomposition heuristic, this means choosing the rover that has the shortest path after including a visit to the new location. For the move heuristic, the new science activity is moved to a time between two existing science activities, which creates a new path shorter than any other possible new path.

This approach has several advantages and disadvantages. One major advantage is that the planning process is conceptually simplified. All commands are sequenced together, allowing any interactions to be easily checked and planned for. Also, planning tends to be computationally expensive and thus requires significant

computational power (e.g., a powerful processor). Missions may have processing power available at one site, but little at other sites (e.g., rovers). On the other hand, a centralized planner would be less desirable for a mission with evenly distributed processing power.

A major disadvantage becomes visible when the rovers' environment is somewhat unpredictable. Here the central planner will also have to monitor execution in order to replan activities in response to unexpected failures or fortuitous events. This will involve continuously transmitting large amounts of data to and from the master agent. Finally, this approach has a single point of failure. If the agent running the planner is rendered inoperable, remote planning will not be possible, and command sequences will need to be uploaded from the ground.

3.2. Central Goal Allocation with Distributed Planning

To support more advanced missions with multiple autonomous rovers, we need to consider distributed planning. This would include rovers planning for themselves, and for other rovers. If there is a slow communication link between rovers, or between rover and lander, a planner may be useful on each rover. This would eliminate the need to constantly transmit monitoring information across the slow communication link. By balancing the workload, distributed planning can also be helpful when individual computing resources are limited. Distributed planning is especially difficult, however, when rovers do cooperative or coordinated activities with shared resources. This would include, for example, several rovers communicating with a single lander.

As shown in figure 5, one approach to distributed planning is to include one planner for each agent, in addition to a central planner. The central planner develops an abstract plan for all agents, 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. When planning with shared resources, aggregate resources are divided equally among the agents that use the resource. In other words, for N agents, each agent models the resource with a capacity that is 1/N of the total capacity. For atomic resources, the availability is time-sliced among the agents. Each agent has the resource available for 1/N of the total time. This guarantees that the resulting set of plans will not have conflicts, even among interacting activities.

This approach also has its advantages and disadvantages. The obvious advantage is that the planning process is distributed across multiple processors. This reduces the workload on any one agent and allows

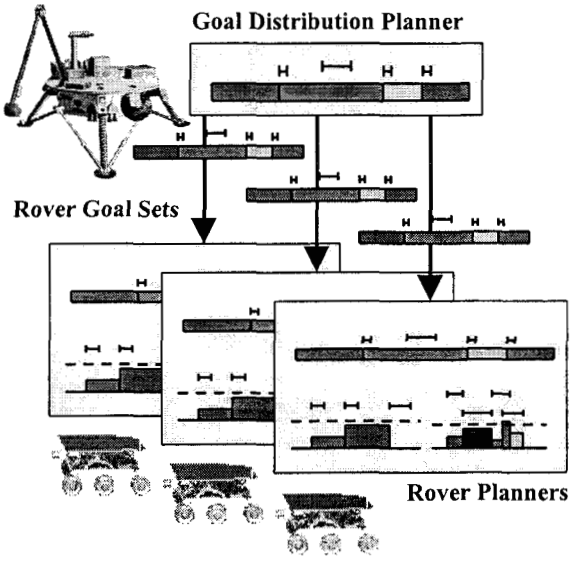

Figure 5. Distributed Planning

planning to be done in parallel. Another major advantage is faster reaction time with less communications. With a planner onboard the rovers, there is a tight loop between planning and execution. This allows shorter turn-around times from execution failures to command sequence updates, which in turn decreases rover idle time. Also, the rovers only have to transmit subsets of their status information, and the central goal allocation planner only transmits smaller abstract plans to the rovers. The only time when the central goal allocation planner has to replan occurs when a local planner runs into a situation that it cannot resolve.

The major disadvantage of this approach stems from the partitioning of goals and resources from the master to the slaves. Once the goals have been assigned, there is no way for them to be reassigned to different rovers. In addition, the equal division of shared resources is an oversimplification. One rover may need a disproportionate amount of a particular resource. This type of resource division limits the set of possible solutions, possibly forcing plans to be sub-optimal.

3.3. Contract Net Protocol

At its extreme, migrating the planning/scheduling process onto the rovers leaves a central auctioneer to distribute goals, and the rovers use planning/scheduling to determine appropriate bids for each goal as it arises. This approach is an instance of the *contract net protocol* (Smith 1980, Sandholm 1993) – a commonly used coordination paradigm within the distributed artificial intelligence community. Within a contract net protocol, a manager announces a task to a set of contractors, each contractor bids for it, and the manager awards the task to the contractor with the best bid.

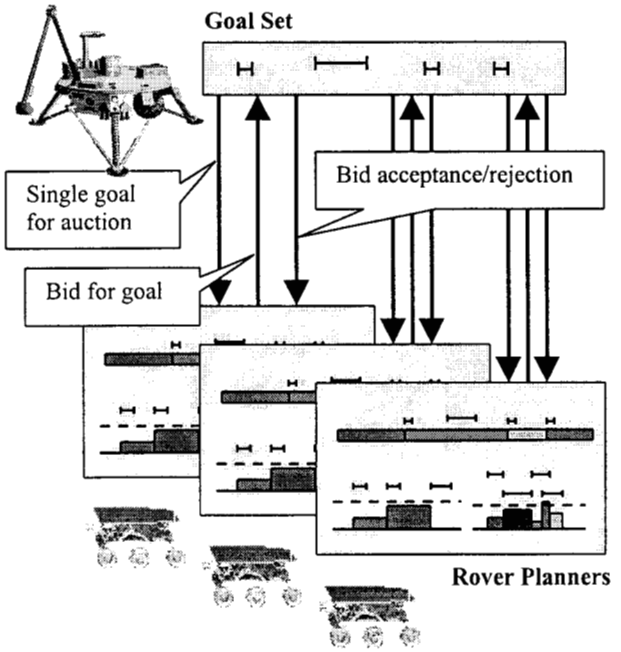

Figure 6. Contract Net Approach

As shown in figure 6, implementing the greedy insertion MTSP algorithm using a contract net protocol involves making the lander take abstract tasks and incrementally transmit them to each rover. Upon receiving a task, a rover uses an ASPEN process with the MTSP heuristics to try to fit the task into its current schedule. Upon succeeding, a rover bids the distance it would travel upon including the new task. Rovers that fail to insert the task within a time limit do not participate in the auction. Upon receiving all bids, the lander awards the task to the rover with the smallest bid. The rovers bid the total distance in order to minimize the maximum distance traveled by any rover. Bidding the incremental distance would bias the system toward solutions that minimize the sum of the travel distances.

This approach has many of the centralized goal allocation algorithm's advantages and disadvantages. Once again, the planners on the rovers facilitate tight feedback between planning and execution without high communications overhead, and partitioning the shared resources on the lander leads to sub-optimal plans. The one difference between the decentralized planning approaches involves the information used to partition the goals. Where the previous approach ignored resources on the rovers and partitioned the goals strictly based on expected path distances, the contract net approach partitioned goals based on path distances after taking other rover resources into account. This change comes at the cost of each rover's having to repair its schedule many more times in order to compute intermediate path distances for partitioning.

4. Comparisons

The three approaches presented in this paper for coordinating multiple agents have a number of functional differences. In addition, these approaches were empirically evaluated using a geological scenario for a number of different metrics. In this section we describe each of these metrics and present the empirical results gathered for each approach.

4.1. Functional Comparison

One main functional difference between approaches is that both the distributed planning approach and contract net approach can take advantage of parallel processing while the centralized planning approach cannot. For the distributed planner, once goals have been allocated to the individual agents (rovers), their planners can run in parallel. Similarly, in the contract net approach, the bidding process can utilize parallel computation by the individual planners to compute the cost of achieving an additional goal on each rover.

Another functional comparison is the number of communications required in each direction (lander to rover (LR), rover to lander (RL)) for each planning cycle for N rovers and G goals. In these computations presume that the centralized computations are occurring on the lander. The centralized planning approach requires N RL and N LR communications for respectively collecting the individual rovers initial states and distributing the resultant plans. The distributed planning approach also requires N RL communications (to denote initial rover locations) and N LR communications to distribute goal sets. The contract net approach requires G iterations, where each iteration consists N LR communications (to send out a new goal) and N RL communications (to respond with a new plan cost). Note that the message sent for a complete plan (as required by the completely centralized approach) is likely to be a longer message than for an initial state (as required by the centralized approach) or goal set (as required by the distributed planning approach). These messages are likely in turn to be longer than messages for a goal or a plan's cost (required by the contract net approach), or initial state (centralized).

Another functional difference is the degree of autonomy offered by an individual rover with respect to possible replanning. In the centralized planner approach, a failure by a rover that cannot plan would require communication with the central planner before resuming execution. In the case of the distributed planning or contract net approach, if the failure could be planned around locally by the failing rover, such communication would not be necessary.

4.2. Empirical Comparisons

In order to test these different approaches, problems were generated from the geological scenario described in

Section 2 of this paper. In this scenario, science goals are for gathering spectral data at different locations with the overall goal of classifying different rock types. These goals are generated using a machine learning clustering algorithm. To produce a good testing set, 10 different (random) rockscapes were used. For each rockscape the following steps were taken. First, the three rovers are placed in different sections of the landscape. Then from these initial locations each rover is asked to perform a set of panoramic spectral readings to gather initial data about the surrounding area. Based on this data the clustering system begins to form rock clusters and requests additional data readings at particular locations to help better discriminate between the different clusters (or rock types). This process was run for 20 iterations on each rockscape (i.e. 200 total iterations), where an iteration consists of the clustering algorithm generating a set of goals and data for those goals being gathered and used to further refine the rock clusters. In each iteration 12 goals for different spectral readings are produced. Thus for each rockscape 120 different science goals were produced.

Using these goal sets, we then ran 10 trials for each of the three architectures. In these tests we calculated the following statistics (averaged over 200 iterations):

- Number of goals achieved (max of 12 per iteration)
- Average distance traveled per goal achieved
- Computation time to generate plans (sum and makespan)

Planning time was calculated in two modes for distributed planning approach and the contract net protocol approach. One mode utilized make-span where it was assumed these methods could be run using parallel processors. The other mode utilized cumulative planning time; thus for the distributed planning approach planning time for the master and each rover is summed and for the contract net approach, the time for each auction summed.

The results of these experiments are shown in Table 1. The contract net approach outperformed the centralized and distributed planning approaches in terms of number of goals achieved. This is because the contract net approach allows the individual planners to each try to achieve every goal – a goal will be deleted only if all rovers cannot plan for it. In the distributed planner case the goals are distributed before planning attempts are made, if the assigned rover cannot achieve a goal, the other rovers do not have a chance to try. In the centralized approach, the planner may delete the goal (with a certain probability) if it is having trouble fitting it into the global plan. Reducing this probability would increase the number of goals achieved but would also increase the CPU cost. The average distance traveled per goal is comparable for all of the approaches. Compared to the centralized approach, the distributed planning approach incurs a greater cost in total computation, but the average makespan of the CPU time is lower because it can construct individual plans in parallel. The contract net approach uses considerably more CPU time because it invokes each of the individual planners G times for G overall goals. As stated previously, this is likely why the contract net achieves more goals on average.

	Ave # Goals Achieved	Ave Dist Travel / Goal	Σ CPU Time	Makespan CPU Time
Centralized Planner	8.31	128.7	18.52	18.52
Distributed Planner	8.22	121.2	25.07	14.11
Contract Net	11.75	127.8	155.89	85.65

Table 1: Empirical Comparison of Coordination Methods

5. Related Work

While there is a large literature on cooperating robots, most of it focuses on behavioral approaches that do not explicitly reason about partitioning goals and planning courses of action. Two notable exceptions are GRAMMPS (Bumitt & Stentz 1998) and MARS (Fischer *et al.* 1995). GRAMMPS is a system coordinating multiple mobile robots visiting locations in cluttered partially-known environments. This system shares quite a bit similarity with our central goal allocation with distributed planning architecture. They both solve an MTSP problem to distribute targets, and they both have low level planners on each mobile robot. The difference involves our focusing on multiple resources and exogenous events while their focus was on path planning while learning a terrain. Also, GRAMMPS uses simulated annealing where we use a greedy approach to solving the MTSP problem.

MARS on the other hand is a cooperative transportation scheduling system that shares many similarities with our contract net approach. Once again the differences involve our focus on multiple resources and exogenous events. Also, the transportation agents bid how much it costs to add a goal to its path. This resulted in minimizing the total distance traveled by all agents. Our rovers bid the total path length after inserting the goal. This difference made our rovers spread out the goals to minimize the maximum distance traveled by any one rover. Finally, MARS also provides a "stock market" for secondary auctions after the initial assignment of goals. Including this facility while reasoning about multiple resources is a future research direction.

6. Conclusions

This paper has described and evaluates three methods for coordinating multiple agents. These agents interact in two ways. First, they are able to work together to achieve a common pool of goals which would require greater time to achieve by any one of the agents operating independently. Second, the agents share resources that are required by the

actions required to accomplish the goals. The first coordination method described is a centralized scheme in which all of the coordination is done at a central location and the agents have no autonomy at the planning level. The second method performs goal allocation using a centralized heuristic planner and (distributed) planners for the individual agents perform detailed planning. The third method uses a contract net protocol to allocate goals and to (distributed) planners for the individual agents perform detailed planning. We compare these approaches and empirically evaluate them using a geological science scenario in which multiple rovers are used to sample spectra of rocks on Mars.

Acknowledgements

This work was performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration. Portions of this work were supported by the Autonomy Technology Program, managed by Dr. Richard Doyle and with Melvin Montemerlo as the headquarters program executive, NASA Code SM.

References

B. L. Bumitt and A. Stentz 1998. GRAMMPS: A Generalized Mission Planner for Multiple Mobile Robots In Unstructured Environments. In *Proceedings of ICRA-98*.

D. Cook, P. Gmystrasiewicz, and L. Holder 1996. Decision-Theoretic Cooperative Sensor Planning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(18).

T. Estlin, S. Hayati, A. Jain, J. Yen, G. Rabideau, R. Castano, R. Petras, S. Peters, D. Decoste, E. Tunstel, S. Chien, E. Mjolsness, R. Steele, D. Mutz, A. Gray, T. Mann 1999. An Integrated Architecture for Cooperating Rovers. In *Proceedings of the International Symposium on Artificial Intelligence Robotics and Automation in Space (ISAIRAS)*, Noordwijk, The Netherlands.

K. Fischer, J. Müller, M. Pischel, and D. Schier 1995, "A Model For Cooperative Transportation Scheduling," in *Proceedings of the First International Conference on Multi-Agent Systems*. San Francisco, CA.

A. Fukunaga, G. Rabideau, S. Chien, D. Yan 1997. Towards an Application Framework for Automated Planning and Scheduling. In *Proceedings of the 1997 International Symposium on Artificial Intelligence, Robotics and Automation for Space*, Tokyo Japan.

J. Hagopian, T. Maxwell, and T. Reed 1994. A Distributed Planning Concept for Space Station Payload Operations. *Third Symposium on Space Mission Operations and Ground Data Systems*, Greenbelt, MD.

D. Hochbaum 1997. *Approximation Algorithms for NP-hard Problems*, PWS Publishing Company.

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

JPL 1999. http://www.jpl.nasa.gov/missions/

M. Mataric 1995. Issues and Approaches in the Design of Collective Autonomous Agents. *Robotics and Autonomous Systems*, 16 (2-4). pp. 321-331.

J. Müller 1996, *The Design of Intelligent Agents, A Layered Approach*, Lecture Notes in Artificial Intelligence, Springer-Verlag.

L. Parker 1998. ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation. *IEEE Transactions on Robotics and Automation*, 14 (2).

G. Rabideau, S. Chien, P. Backes, G. Chalfant, and K. Tso 1999. A Step Towards an Autonomous Planetary Rover. *Space Technology and Applications International Forum (STAIF)*, Albuquerque, NM.

T. Sandholm. 1993. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. In *Proceedings of AAAI-93*.

G. Smith. 1980. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, 29(12).