From Rovers to Orbiters: Continuous Task-Distribution-Based Coordination

Anthony Barrett
Jet Propulsion Lab, M/S 126-347
California Institute of Technology
4800 Oak Grove Drive, Pasadena, CA 91109-8099
anthony.barrett@jpl.nasa.gov

Abstract

This paper compares and contrasts several coordination schemes for a system that continuously plans to control collections of rovers or spacecraft using collective mission goals instead of goals or command sequences for each spacecraft. A collection of self-commanding robotic systems would autonomously coordinate itself to satisfy high level science and engineering goals in a changing partially understood environment – making feasible the operation of tens or even a hundred spacecraft.

1. Introduction

While explicitly commanding a spacecraft via low level command sequences has worked spectacularly on previous NASA missions, there are limitations deriving from communications restrictions – scheduling time to communicate with a particular spacecraft involves competing with other projects due to the limited number of deep space network antennae. This implies that a spacecraft can spend a long time just waiting whenever a command sequence fails. This is one reason why the New Millennium program has an objective to migrate parts of mission control tasks onboard a spacecraft to reduce wait time by making spacecraft more robust [Muscettola *et al.* 1997]. The migrated software is called a "remote agent" and can be partitioned into 4 components:

- a mission manager to generate science goals,
- a planner/scheduler to turn goals into executable activities through reasoning about expected future situations,
- an executive/diagnostician to initiate and maintain activity execution while interpreting sensed events through reasoning about past and present situations, and
- a conventional reactive controller to interface with the spacecraft to implement an activity's execution.

In addition to needing remote planning and execution for isolated spacecraft, a trend toward multiple-spacecraft missions points to the need for remote distributed planning and execution. The past few years have seen missions with growing numbers of probes. Pathfinder has its rover (Sojourner), Cassini has its Huygens lander, and Cluster II is scheduled to launch in 2000 and has 4 spacecraft for multi-point magnetosphere plasma measurements. This trend is expected to continue to progressively larger fleets. For example, one proposed interferometer mission [Mettler& Milman 1996] would have 18 spacecraft flying in formation in order to detect earth-sized planets orbiting other stars. Another proposed mission involves 44 to 104 spacecraft in Earth orbit to measure global phenomena within the magnetosphere.

This paper compares and contrasts 3 ways to distribute a planner/scheduler amongst a population of spacecraft or rovers that have separate executive/diagnosticians and reactive controllers. The first places the planner/scheduler on a single platform that remotely commands the others. The second is more distributed in that it replicates a planner across the population to let each platform plan its own activities, but a single platform handles goal distribution. The last approach advertises all goals and lets each platform bid for a goal based on how well its local planner can satisfy the goal given local information. These approaches delineate a space of approaches where the platform that distributes tasks maintains progressively less information on the entire constellation.

This paper's sections subsequently describe 3 thought experiments for multi-platform missions that motivate 8 performance metrics for evaluating approaches toward continuous task-distribution-based coordination, compare and contrast 3 coordination methods, discuss related work, and finally conclude.

2. Multi-Platform Thought Experiments

In order to focus this discussion on distributed autonomy in space, consider different types of future multi-platform missions. There are 4 kinds of such missions depending on the reason for proposing multiple platforms:

- improved coverage when observing/exploring large areas (like the number of identical small satellites with scatterometers proposed for the Ocean Surface Wind Measurement Program (EOS-5));
- specialized probes with explicitly separate science objectives (like Cassini with Huygens and Pathfinder with Sojourner);

- multi-point in-situ sensing for observing large scale phenomena that are only detectable with multiple spatially separated in-situ sensors (like for observing global magnetospheric phenomena with spatially separated plasma sensors in the Magnetospheric Multi Scale or Cluster II missions); and
- building large synthetic aperture sensors with many small spatially separated sensors for imaging very remote targets (like Constellation-X, Terrestrial Planet Finder, and TechSat-21).

These reasons for having multiple platforms in a mission are not exclusive. For instance, the Air Force's TechSat-21 mission concept [Martin&Stallard 1999] involves a constellation of clusters of platforms. Each cluster forms a synthetic aperture for radar sensing, and the number of clusters depends on the desired global coverage.

2.1. Coordinating Task Distribution

In missions where each probe performs its task in isolation, the difference between an autonomous multi-platform mission and many autonomous single platform missions involves distributing tasks to the different platforms. While the task distribution for multiple autonomous single platform missions is determined on the ground, an autonomous multi-platform mission can distribute and redistribute tasks remotely. This feature improves both distribution quality and robustness by letting the spacecraft use local information to optimize the initial task distribution and to redistribute tasks when a spacecraft suffers an anomaly, unexpectedly finishes a task early, or detects an unanticipated science opportunity.

As an example of coordinated autonomous task distribution, consider multiple rovers surveying the rocks in an area on Mars using MISUS [Estlin *et al.* 1999]. In this system a Mars lander manages a population of rovers by analyzing data from past observations, determining new observations, assigning observation goals to rovers, and collecting data as each rover moves from rock to rock and performs its experiments in isolation (fig. 1). This system's objective is to maximize science return while minimizing the execution time of the most heavily tasked rover.

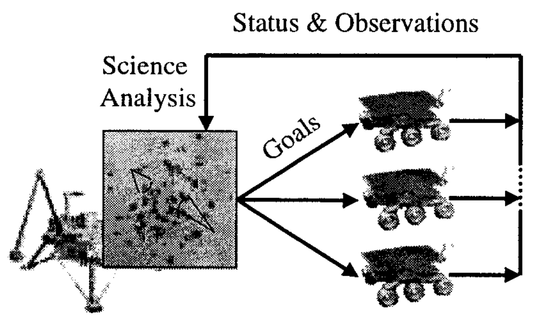


FIG 1: Coordinating multiple rovers with MISUS

While MISUS focused on a multi-rover scenario, much of the developed infrastructure applies to any multi-platform mission with a number of identical platforms that operate in isolation. This is includes most of the improved coverage and specialized probe classes of missions. As an example of this generalization, consider replacing the Mars lander and the rover population with a ground station and a constellation of RADARSATs like the one illustrated in figure 2. Currently the Canada Centre for Remote Sensing manages a single RADARSAT [CCRS 1998] in a sun-synchronous orbit. This satellite can observe any location around the poles on a daily basis and any location around the equator in 6 days or less. To decrease the equatorial delay time, consider replacing the single satellite with 6 equally spaced RADARSATs. The resultant constellation decreases the equatorial delay time to one day. A system like MISUS could manage this constellation since each of the 6 satellites operates independently of the others.

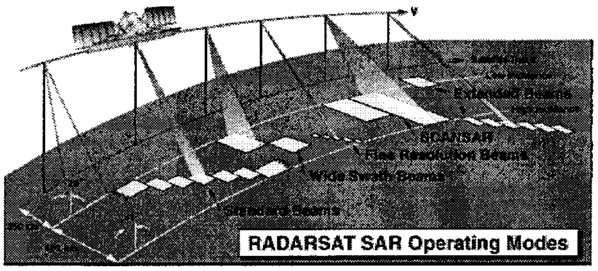


FIG 2: Operating modes for one of a constellation of radar satellites

2.2. Coordinating Task Execution

The multi-point in-situ sensing and large synthetic aperture missions differ operationally from the other 2 classes in that the separate spacecraft do not operate in isolation. For instance consider the Air Force's TechSat-21 mission concept (fig. 3). TechSat-21 involves a constellation of clusters of spacecraft. While each cluster can function in

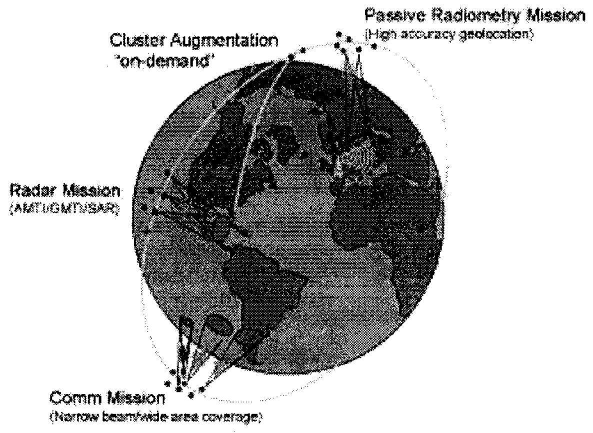

Passive Radiometry Mission
(High accuracy geolocation)
Cluster Augmentation
"on-demand"
Radar Mission
(AMTI/GMTI/SAR)
Comm Mission
(Narrow beam/wide area coverage)

FIG 3: Operating modes for clusters in TechSat-21

isolation, spacecraft within a cluster have tightly coordinated activities. These activities include tight beam communications, synthetic aperture radar (SAR), and geolocation.

In many respects coordinated task execution is easier than coordinated task distribution. For the smaller missions designating a master spacecraft that commands the other (slave) spacecraft as though they were physically attached solves this problem, but bandwidth restrictions keep this approach from scale with either the number of slaves or the complexity of each slave.

2.3. Autonomy Architectures

In an earlier paper [Barrett 1999], I described 3 different autonomy architectures for a constellation of spacecraft involving leaders, followers, and slaves. Here I expand this taxonomy to also include contractors. The number of autonomy modules on a spacecraft determines which of the 4 classes it falls into:

- a *slave* has no modules and is tele-operated by the reactive control module of another nearby spacecraft;
- a *follower* has both an executive/diagnostician and a reactive controller (like many existing spacecraft);
- an *contractor* has a follower's components and a planner/scheduler to optimize local activities (like DS1's remote agent experiment); and
- a *leader* has all four components.

With these 4 classes, we can define a multi-platform mission's autonomy architecture by stating the class of each platform, and how the collection of platforms coordinate their activity. In terms of MISUS, the architecture consists of having the lander lead, and letting the rovers act as followers or contractors depending on the desired local autonomy.

Given a multi-platform mission, there are two sets of metrics for evaluating the acceptability of autonomy software. The first set motivates minimizing the amount of remote autonomy and has 4 metrics:

- the amount of explicit control an operator has over the constellation's activities,
- the feasible accuracy of modeling the constellation's activities on the ground,
- the autonomy software's testability, and
- the amount of needed onboard computing power.

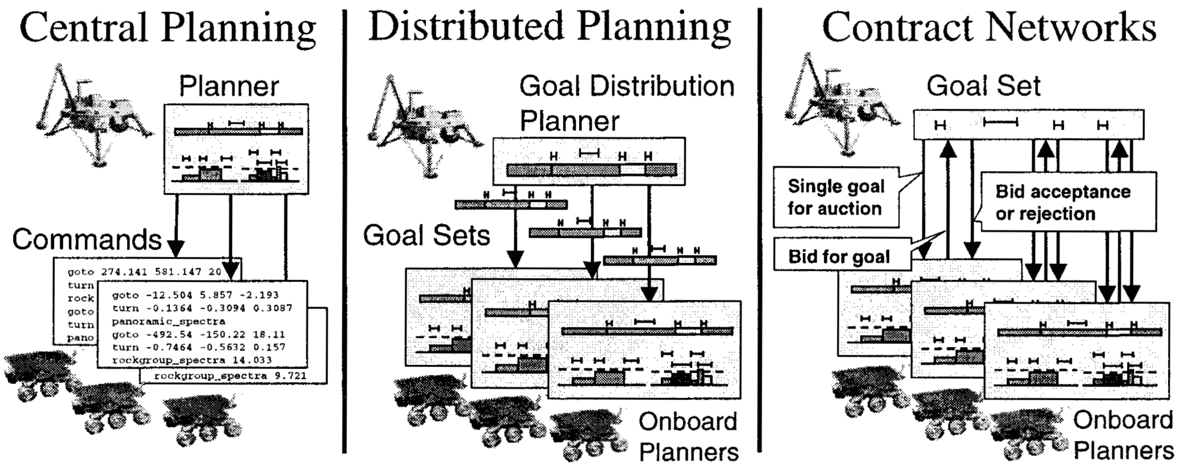
While the first set of metrics tend to be maximized by limiting the amount of autonomy on a constellation, the second set of 4 evaluation metrics are maximized by increasing the amount or remote autonomy:

- the platforms' event response time,
- the required bandwidth between platforms and to Earth,
- the quality of the downlinked data, and
- the functional redundancy.

3. Coordinating Multiple Planners

In [Rabideau *et al.* 1999], others and I compared 3 methods for coordinating a population of rovers from a central lander in the MISUS scenario (figure 4). We used central planning to manage a population with a leader-follower architecture, where the leader generates plans that are subsequently executed by the followers. In order to assure each plan's correctness, the lander needs to acquire large amounts of state information on the rovers to appropriately determine if they can execute their plans.

Distributed planning reduces the amount of needed state information by using a goal distribution planner. This planner takes a subset of the rovers' collective state information with less precise models of the rovers, and it produces an abstract plan with enough detail to determine how to distribute the goal activities among the rovers. The lander then transmits goals to the appropriate rovers.



```
goto 274.141 581.147 20
turn
rock
goto
turn
pano
```

```
goto -12.504 5.857 -2.193
turn -0.1364 -0.3094 0.3087
panoramic_spectra
goto -492.54 -150.22 18.11
turn -0.7464 -0.5632 0.157
rockgroup_spectra 14.033
```

```
rockgroup_spectra 9.721
```

FIG 4. Approaches to task distribution based coordination

Another way to migrate planner/schedulers onto the rovers uses a central auctioneer to distribute goals, and the rovers use onboard planners with local state information to determine appropriate bids for each goal as it arises. This approach is an instance of the *contract network protocol* [Smith 1980, Sandholm 1993] – a commonly used coordination paradigm within the distributed artificial intelligence community. Within a contract net protocol, a leader announces a task to a set of contractors, each contractor bids for it, and the leader awards the task to the contractor with the best bid.

3.1. Central Planning

The simplest way to extend single-platform spacecraft autonomy research to autonomous multi-platform missions involves using a master/slave approach where a single leader performs all autonomy reasoning. The slaves only transmit sensor values to the leader and forward control signals received from the leader's reactive controller to their appropriate local devices. In this way all spacecraft are treated as a single multi-platform spacecraft.

Both single and multiple platform autonomous spacecraft must respond to a (somewhat) dynamic, unpredictable environment. In terms of high-level, goal-oriented activity, a planner needs to modify spacecraft sequences to account for fortuitous events such as observations completing early and setbacks such as a failure to acquire a guide star for a science observation.

The need to rapidly respond to unexpected events motivates continuous planning, an approach where a planner continually updates a sequence in light of changing operating context. In such an operations mode, a planner would accept and respond to activity and state updates on a one to ten second time scale. CASPER [Chien *et al.* 1999] is an example of a continuous planner based on a heuristic iterative repair approach toward planning [Zweben *et al.* 1994, Fukunaga *et al.* 1997]. This approach takes a complete plan at some level of abstraction and manipulates its actions to repair detected flaws. Example flaws would involve an action being too abstract to execute or many simultaneous actions with conflicting resource needs.

Making a heuristic iterative repair planner continuous within a planner/scheduler module results in figure 5's algorithm. The first line assures that the PROJECTION variable always reflects how the state of a rover, or a spacecraft, should evolve as its plan executes, and the last line causes this execution by passing near-term activities to the executive/diagnostician.

The expected state evolution changes as a plan gets new goal activities and the perceived state diverges from expectations. This divergence is caused by unexpected exogenous events and activities having unexpected outcomes. Since a planning model can only approximate the reality experienced during execution, these unexpected state changes can always to happen.

```
Given: a PLAN with multiple activities
       a PROJECTION of PLAN into the future
1. Revise PROJECTION using the currently
   perceived state and new goal activities from the
   mission manager.
2. Heuristically choose a plan flaw found in
   PROJECTION.
3. Heuristically choose a flaw repair method.
4. Use method to alter PLAN & PROJECTION.
5. Release relevant near-term activities in PLAN to
   the real-time system.
6. Go to 1.
```

FIG 5. Continuous planning using heuristic iterative repair

At any moment the projection can detect flaws in a local plan, and lines 2 through 4 select and apply repair methods to fix these flaws. For instance, a satellite's observation activity can take an unexpectedly long time to complete. Depending on the delay, a subsequent observation may be impossible due to the target being too far behind the satellite when the observation starts. A repair method might fix the flaw by rescheduling the observation at a later time.

With respect to our evaluation metrics, using a continuous planner with a master/slave approach toward multi-platform coordination facilitates allowing a variable amount of remote autonomy. At one extreme the continuous planner is given low-level command sequences and can only apply a go-to-safe-mode repair method upon detecting a flaw. This extreme maximizes the first set of metrics. Another extreme reduces the first set of metrics while improving the second set. Here the planner is only given a set of abstract activities and uses local information and heuristics improve event response time and the quality of downlinked data. While functional redundancy and inter-platform bandwidth are unaffected by moving from one extreme to another, turning the slaves into followers increases redundancy and reduces bandwidth. Due to how easily this change can degrade the event response time, turning slaves into followers is an active research topic in the multi-agent research community [Tambe 1997].

3.2. Distributed Planning

Turning followers into contractors raises issues regarding how to coordinate multiple planners. In distributed planning, this coordination is achieved through using a continuous goal distribution planner on one platform, and this planner continuously manages the distribution of goals based on continuously updated partial information on the other platforms. For instance, the distribution planner might model rovers in a multi-rover scenario as points on a plane where each rover can travel in a straight line from one goal activity's observation target to another's.

With this abstract characterization, the distribution planning problem becomes a Multiple-Traveling Salesman Problem (MTSP) [Johnson&McGeoch 1997] where the members of a sales team must collectively visit each of a set of cities and the maximum traveling time of the salesmen is minimized. While this is a NP-Complete problem, there are fast greedy approaches that find slightly sub-optimal solutions. By encoding one of these approaches into our distribution planner, the lander can both determine how to distribute the goal activities and provide a rough estimate on the order in which a rover should visit its targets to perform the goal activities.

With respect to our evaluation metrics, distributed planning facilitates variable autonomy both with the ground and across the platforms. Minimizing autonomy across platforms involves making the distribution planner use full information and generate low level action sequences for the other platforms, which can only execute their actions. This restriction turns distributed planning into the previously evaluated central planning approach.

Maximizing autonomy on the contractor platforms has the same effects as maximizing autonomy for the central planner, but also adds a reduction to inter-platform bandwidth needs. The lead platform no longer needs to maintain full state information, and each platform's planner can locally respond to events without informing the leader. Now a contractor can resolve a flaw by either quietly shuffling its local activities or reporting failure to the leader upon deleting a local activity. This quiet shuffle reduces bandwidth needs while failure reporting facilitates moving activities between platforms via the leader's continuously repairing its goal distribution plan.

3.3. Contract Networks

Minimizing the amount of continuously updated partial contractor information on the leader results in taking a contract network approach toward coordinating multiple planners. Here a leader advertises each goal and contractors bid on the goals based on local information. To respond to an unexpected event, a contractor will either quietly shuffle its activities or delete a local activity and report failure to the leader. Upon hearing of a failure, the leader can re-advertise the failed goal for auction. Notice that there is no need for continuously updated partial contractor information – the leader does not need to know anything about the contractors to auction a goal.

As shown in figure 4, using a contract net protocol to implement a greedy solution to the MTSP involves making the lander take goal activities and incrementally advertise them to all rovers. Upon receiving a task, a rover uses an onboard planner to try to fit a solution to the goal activity into its current schedule. Upon succeeding, a rover bids its total projected travel distance upon including the new observation. 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. By bidding the total distance the rovers minimize the maximum rover travel distance – an MTSP solution.

With respect to our evaluation metrics, letting an operator restrict the platforms that can bid for certain activities results in a system with variable autonomy. At one extreme the operator can specify a low level activity sequence for each platform, and at the other the leader gets a set of high level goals that can go to any platform.

As before, the first extreme scores best on the autonomy minimization metrics and the second scores best on the autonomy maximization metrics. While this approach has lower inter-platform bandwidth needs than the other approaches, it has more computational overhead and assumes a greedy approach toward optimization.

4. Related Work

While there is a large literature on cooperating robots, most focuses on behavioral approaches that do not explicitly reason about partitioning goals and planning courses of action. Three notable exceptions are GRAMMPS [Bumitt&Stentz 1998], MARS [Fischer *et al.* 1995], and RETSINA [Paolucci *et al.* 1999]. 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 for rovers. Both systems solve an MTSP problem to distribute targets, and both have low level planners on each mobile robot, but GRAMMPS focuses on path planning while learning a terrain instead of focussing on resources and exogenous events.

MARS on the other hand is a cooperative transportation scheduling system that shares many similarities with the contract net approach. Once again the differences involve a focus on multiple resources, exogenous events, and variable autonomy.

Finally RETSINA uses peer-to-peer coordination with an HTN planner for local planning. While the use of heuristic iterative repair here points to one difference between the approaches. The main difference involves RETSINA's not modeling known exogenous events and not providing default mechanisms for initially distributing goals and transferring goals to resolve execution failures.

5. Conclusions

This paper compared and contrasted 3 continuous task-distribution-based coordination schemes for commanding multiple platforms with collective goals instead of goals or command sequences for each platform: central planning, distributed planning, and contract networks. All schemes supported variable autonomy and were evaluated with respect to 8 different metrics. At the lowest autonomy setting, all schemes devolved into commanding the platforms with a separate low level sequence for each, and at the highest autonomy setting the schemes differed primarily in terms of needed onboard computing, inter-platform bandwidth, and redundancy. While central planning kept all computing on the leader, distributed planning spread the computing overhead across all platforms. The result was a decrease in inter-platform bandwidth needs and an increase in redundancy with an unchanging total computing overhead. Contract networks further improved the bandwidth needs and redundancy, but this scheme also increased the total computing overhead by letting each platform see and bid for each goal.

Reasoning about incremental autonomy for distributed planning and contract networks results in a realization that these approaches toward coordinating multiple planner/schedulers can be combined. The resultant approach would used a goal distribution planner, but would only collect enough information to limit the number of platforms that participate in an auction. One avenue for future work involves building a coordination mechanism that spans the space between contract networks and distributed planning. Another future research avenue involves generating joint activities for multiple spacecraft/rovers to collectively satisfy and would extend our system to handle constellations of clusters of platforms (in TechSat-21). Finally, a third research direction involves making the rovers/orbiters compete for shared resources, like communications opportunities.

Acknowledgements

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

References

[Barrett 1999] A. Barrett, "Autonomy Architectures for a Constellation of Spacecraft," International Symposium on Artificial Intelligence Robotics and Automation in Space (ISAIRAS), Noordwijk, The Netherlands, June 1999.

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

[CCRS 1998] Canadian Centre for Remote Sensing, http://www.ccrs.nrcan.gc.ca/ccrs/tekrd/radarsat/specs/radovere.html.

[Chien *et al.* 1999] S. Chien, R. Knight, A. Stechert, R. Sherwood, and G. Rabideau, "Using Iterative Repair to Increase the Responsiveness of Planning and Scheduling for Autonomous Spacecraft," IJCAI99 Workshop on Scheduling and Planning meet Real-time Monitoring in a Dynamic and Uncertain World, Stockholm, Sweden, August 1999.

[Estlin *et al.* 1999] T. Estlin, T. Mann, A. Gray, G. Rabideau, R. Castano, S. Chien and E. Mjolsness, "An Integrated System for Multi-Rover Scientific Exploration," in *Proceedings of AAAI-99.*

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

[Fukunaga *et al.* 1997] 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 Intell-igence, Robotics and Automation for Space*, Tokyo Japan.

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

[Martin&Stallard 1999] M. Martin and M. Stallard, "Distributed Satellite Missions and Technologies – The TechSat 21 Program," AIAA 1999 Space Technology Conference, Albuquerque, NM, September 1999.

[Mettler&Milman 1996] E. Mettler and M. Milman. "Space Interferometer Constellation: Formation Maneuv-ering and Control Architecture," In SPIE Denver '96 Symposium.

[Muscettola *et al.* 1997] N. Muscettola, *et al.* "On-Board Planning for New Millennium Deep Space One Autonomy," in Proceedings of IEEE Aerospace Conference 1997.

[Paolucci *et al.* 1999] M. Paolucci, D. Kalp, A. Pannu, O. Shehory, and K. Sycara, "A Planning Component for RETSINA Agents," Lecture Notes in Artificial Intelligence, Intelligent Agents VI. M. Wooldridge and Y. Lesperance (Eds.), forthcoming.

[Rabideau *et al.* 1999] G. Rabideau, T. Estlin, S. Chien, A. Barrett, "A Comparison of Coordinated Planning Methods for Cooperating Rovers," AIAA 1999 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 AAAI-93.*

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

[Tambe 97] M. Tambe, "Towards Flexible Teamwork," *Journal of Artificial Intelligence Research*, 7:83-124.

[Zweben *et al.* 1994] M. Zweben, B. Daun, E. Davis, and M. Deale, "Scheduling and Rescheduling with Iterative Repair," in *Intelligent Scheduling*, Morgan Kaufman, San Francisco, 1994.