Market-Based Multirobot Coordination For Complex Space Applications

M. Bernardine Dias, Dani Goldberg, and Anthony Stentz

Carnegie Mellon University, Pittsburgh, PA, USA
{mbdias, danig, axs}@ri.cmu.edu

Abstract

Multirobot coordination remains a challenging problem with potentially high impact on space applications if solved. The challenge is to enable robots to work together in an intelligent manner to execute a global task. This paper describes a market-based approach suitable for complex multirobot space applications. The market approach has had considerable success in the multirobot coordination domain. Previously published work introduced the market architecture and showed that it is inherently distributed, but can also opportunistically form centralized sub-groups to improve efficiency, and thus approach optimality. This paper focuses on the space application domain, and more specifically, presents simulation results for market-based coordination of a group of heterogeneous robots engaged in information gathering on a Martian outpost.

Introduction

As the limits of space exploration expand, robots play an increasingly important role in mission success and hence are tasked with increasingly difficult operations. Scarce resources, dynamic environments, incomplete information, restricted communication and localization capabilities, high cost of failure, and competing requirements add to the challenge of robotic solutions for the space application domain. A single robot is no longer the best solution for many of these applications; instead, teams of robots are required to coordinate intelligently for successful task execution.

Multirobot solutions are paramount for several reasons. A single robot is not an efficient solution for tasks such as automated construction and mapping/investigation of unknown/hazardous environments. Instead, a team of robots can perform such tasks more efficiently by dividing the task into sub-tasks and executing them concurrently. Furthermore, design constraints can be relaxed since a team can make effective use of specialists designed for a single purpose rather than requiring that a single robot with versatile capabilities be a generalist – capable of performing all tasks, but expert at no tasks. In partially known environments where a single robot would have to rely exclusively on landmarks of some sort for localization, a team could have added benefit from teammates exchanging localization information.

A team generally provides a more robust solution by introducing redundancy, and by eliminating any single point of failure (as long as there is overlap between the robots’ capabilities). A team of robots can also produce a wider variety of solutions than a single robot can, and hence can opportunistically respond to dynamic conditions in more creative and efficient ways. Even if the team does not overlap entirely in terms of specialization, the collective resources of the group can be used in creative ways to solve problems. Thus, robot teams have many advantages over single robots in the space application domain. However, coordinating such a team in an efficient manner remains a challenging problem.

This paper describes a market-based approach suitable for complex space applications. In previous work, we introduced the market architecture (TraderBots) and showed that it is inherently distributed, but can also opportunistically form centralized sub-groups to improve efficiency, and thus approach optimality. Robots are self-interested agents, with the primary goal of maximizing individual profits. The revenue and cost models and rules of engagement are designed so that maximizing individual profit has the benevolent effect of moving the team towards the globally optimal solution. The TraderBots architecture inherits the flexibility of market-based approaches in allowing cooperation and competition to emerge opportunistically.

Next, a space-relevant multirobot scenario is presented and the required characteristics for a successful coordination mechanism are explored. The TraderBots market mechanism is then briefly described and implementation details are presented. The following section presents our experimental Mars rock characterization scenario, as well as results from testing some of the basic parameters of our market mechanism. The paper ends with conclusions, acknowledgements, and references.
Multirobot Space Scenario

A suitable multirobot scenario, relevant to space applications, is the problem of robotic exploration of Mars:
For the foreseeable future, mobile robots will serve as the remote sensors and data collectors for scientists. To create an outpost for such long-term exploration, the robots need to assemble solar power generation stations, map sites, collect science data, and communicate with Earth on a regular basis. In this scenario, on the order of ten robots are sent, many with different capabilities. Some of the robots specialize in heavy moving and lifting, some in science data collection, some in drilling and coring, and some in communication. The rovers have different, but overlapping, capabilities.

The rovers cooperatively search for a location suitable in size and terrain for a base station. Once such a location is found, rovers with appropriate capabilities form several teams to construct the base station capable of housing supplies and generating energy. Two rovers carry parts, such as solar panels, that are too large for a single rover. Complementary capabilities are exploited. Meanwhile, other rovers begin general exploration of the region. To start, several scouting robots (perhaps joined by aerial vehicles) quickly survey the region. Scientists on Earth (and perhaps the rovers themselves) identify sites within the region that have high likelihood to contain interesting science data. Rovers with specialized sensing instruments are sent to investigate. If a particular subtask requires more intensive scrutiny, additional rovers with appropriate capabilities are brought in. Rover failures are addressed by dispatching a rover with diagnostic capabilities. The diagnostic rover can use its cameras to view the failed robot to see if it can be aided in the field, or it may drag the rover back to the base station to be repaired by replacement of failed modules. In the meantime, another robot with the same (or similar) capabilities can be substituted, so as to complete the original task with minimal interruptions. At any given time, different teams of rovers may be involved in exploration, base-station construction/maintenance, and rover diagnosis/repair. Many tasks will be time critical, requiring execution within hard deadlines or synchronization with external events. The teams form dynamically, depending on the task, environment, and capabilities and availability of the various robots to best meet mission requirements over time. The rovers negotiate their individual roles, ensure safety of the group and themselves, and coordinate their precise actions, attempting as a group to avoid unnecessary travel time, to minimize reconfiguration and wait time, and to prefer more reliable alternatives in cases of overlapping capabilities. The challenge is to keep all the robots healthy and busy in appropriate tasks in order to maximize the scientific data collected.

A scenario like this demands high quality performance from a multirobot system. Hence, a general multirobot solution for an application domain such as this must fulfill many requirements. The following characteristics are thought to be an exhaustive list of these requirements:
1. (Robustness): Robust to robot/agent failure, or no single point of failure for the system. This is an important characteristic since many applications rely on continued progress even if some components in the system fail. The space application domain expects that several agents will malfunction or be destroyed during operation, and still require the overall mission to be completed in the best way possible given the remaining resources.
2. (Dynamic Conditions): Opportunistically optimized response to dynamic conditions. This characteristic is desirable in general, and required in some domains. Since the space application domain involves dynamic conditions, the ability to opportunistically optimize the system response to these conditions is necessary for success.
3. (Speed): Quick response to dynamic conditions. Often in dynamic environments, a key to successful task execution is the capability to respond quickly to the dynamic conditions. If information always needs to be channeled to another location for plan modification, conditions can change too rapidly for the planning to keep up.
4. (Extensibility): Easily extendable to accommodate new functionality. A key characteristic
to building a generalized system that can evolve with the needs of the different applications is the ability to easily add and remove functionality as needed. This is identified as extensibility.

5. (Communication): Ability to deal with limited and imperfect communication. In general, many application domains cannot realistically guarantee perfect communication among all agents at all times. Hence, any generalized coordination architecture should be robust to communication failures and limits in range of communication.

6. (Resources): Ability to reason about limited resources. The ability to reason about the limited resources available in a robotic system is very important for optimization purposes.

7. (Task Allocation): Efficient allocation of tasks. A key difficulty in coordinating multiple robots is deciding who does what. Thus, the task allocation mechanism is an important factor in the architectural design.

8. (Heterogeneity): Ability to accommodate heterogeneous teams of robots. Many architectures assume homogeneity for ease of planning. The coordination problem is more difficult if the robots are heterogeneous. A successful architecture will be able to accommodate any team regardless of its homogeneity or heterogeneity.

9. (Roles): Efficient adoption of roles. In many architectures robots are restricted to being able to play only a single role in the team at any given time. Yet, they possess the resources to be able to play more than a single role. Efficient role adoption will enable robots to play as many roles as required at any given time based on resource availability, and also allow robots to change in and out of different roles as conditions change.

10. (New Input): Ability to dynamically handle new tasks, resources, and roles. In many dynamic application domains, the demands on the robotic system can change during operation. Hence, it may become necessary to assign new tasks, change existing tasks, add new resources, or introduce new roles. All of this should be supported by the architecture.

11. (Flexibility): Easily adaptable for different applications. Since different applications will have different requirements, a general architecture will need the ability to be easily reconfigured for the different problems it proposes to solve. Instructions and advice on how to reconfigure the architecture for different applications will also be useful.

12. (Fluidity): Easily able to accommodate the addition/subtraction of robots during operation. Several applications could require the ability to introduce new robots into the system during operation. Similarly, robots can exit or malfunction during task execution. A successful architecture will be able to support such events gracefully.

13. (Learning): On-line adaptation for specific applications. While a generalized system is often more useful, its application to specific domains usually requires some tuning. The ability to tune relevant parameters automatically in an on-line fashion is thus a very attractive feature that can save a lot of effort.

14. (Implementation): Implemented and proven on a physical system. As with any claim, a proven implementation is far more convincing. Moreover, successful implementation of an architecture on a robotic system requires discovering and solving many details that are not always apparent in simulation and software systems.

We are developing a market-based coordination mechanism, TraderBots, which promises to satisfy all of these requirements.

The Market Mechanism: TraderBots

Stentz and Dias [11] first introduced and detailed the philosophy of using a market approach to coordinate multiple robots to cooperatively complete a task, building on the contract net protocol by Smith [10], its extension by Sandholm and Lesser [8], and the general concepts of market-aware agents developed by Wellman and Wurman [13]. This work introduced the methodology of applying market mechanisms to intra-team robot coordination (i.e. in typically non-competitive environments) as opposed to competitive multirobot domains and competitive inter-agent interactions in domains such as e-commerce. Simulation results using this approach were produced by Dias and Stentz [1], and proven robot results were presented by Thayer et al. [12], and Zlot et al. [14]. A few other groups have also published research relevant to market-based multirobot coordination. Golfarelli and Rizzi [6] proposed and implemented a swap-based negotiation protocol for multirobot coordination that restricted negotiations to task-swaps, and Gerkey and Matarić [4] developed the MURDOCH publish/subscribe mechanism that includes a single-round auction for task distribution. Rabideau et al. [7] published a comparison study of three multi-rover coordination mechanisms that included a contract-net-based approach. However, to date, no other group has explored, in detail, a market-based approach to multirobot coordination. A brief summary of our approach is presented next.

Consider a team of robots assembled to perform a particular set of tasks. Consider further, that each robot in the team is modeled as a self-interested

---

1 Note that the name “TraderBots” was recently adopted for our market-based coordination work that started in 1999.

2 For a detailed description of the TraderBots approach please see Dias and Stentz [2].
agent, and the team of robots as an economy. The goal of the team is to complete the tasks successfully while minimizing overall costs. Each robot aims to maximize its individual profit (which often translates to minimizing individual cost where possible); however, since all revenue is derived from satisfying team objectives, the robots’ self-interest equates to doing global good. Moreover, all robots can only increase their profit by eliminating unnecessary waste (i.e. excess cost). Hence, if the global cost is determined by the summation of individual robot costs, each deal made by a robot (note that robots will only make profitable deals) will result in global cost reduction. The competitive element of the robots bidding for different tasks enables the systems to decipher the competing local information of each robot, while the currency exchange provides grounding for the competing local costs in terms of the global value of the tasks being performed.

Revenues, Costs, the Role of Price and the Bidding Process
Appropriate functions are needed to map possible task outcomes onto revenue values and to map possible schemes for performing the task onto cost values. As a team, the goal is to execute some plan such that the overall profit (the excess of revenue over cost), is maximized. Furthermore, these functions must provide a means for distributing the revenue and assessing costs to individual robots. Thus, robots receive revenue and incur costs for accomplishing a specific team-task, but the team’s revenue function is not the only source of income. A robot can also receive revenue from another robot in exchange for goods or services. The price dictates the payment amount for the good or service. A common approach is to bid for a good or service in order to arrive at a mutually acceptable price.

Cooperation vs. Competition
Two robots are cooperative if they have complementary roles; that is, if both robots can make more profit by working together than by working individually. Conversely, two robots are competitive if they have the same role; that is, if the amount of profit that one can make is negatively affected by the presence of the other robot. The flexibility of the market-model allows the robots to cooperate and compete as necessary to accomplish a task.

Self Organization, Learning and Adaptation
Conspicuously absent from the market approach is a rigid, top-down hierarchy. Instead, the robots organize themselves in a way that is mutually beneficial. Since the aggregate profit amassed by the individuals is directly tied to the success of the task, this self-organization yields the best results. Moreover, the robot economy is amenable to learning new behaviors and strategies as it executes its complex global task. An added strength of the market approach is its ability to deal opportunistically with dynamic environments.

Implementation Details
Previously published work includes an implementation of the TraderBots architecture that was developed and tested for a distributed sensing task in a simulated interior environment ([1] and [12]) and also on a group of Pioneer II-DX robots [14]. In this implementation, the market architecture seeks to maximize benefit (information gained) while minimizing costs (in terms of the collective travel distance), thus aiming to maximize utility. The system is robust in that exploration is completely distributed and can still be carried out if some of the colony members lose communications or fail completely. The effectiveness of our approach was demonstrated through successful mapping results obtained with the team of robots. We found that by allowing the robots to negotiate using the market architecture, exploration efficiency was improved by a factor of 3.4 for a four-robot team. An initial implementation of the leader role as a combinatorial exchange was also explored [3]. Results thus far prove the market approach to be highly promising as a multirobot coordination mechanism for mapping and exploration.

This paper focuses on the space application domain, and more specifically, presents simulation results for market-based coordination of a group of heterogeneous robots engaged in information gathering on a Martian outpost. In our current implementation, the market-based, multi-robot planning capability is designed as part of a distributed, layered architecture for multi-robot control and coordination3. More specifically, our architecture is an extension to the traditional three-layered robot architecture that enables robots to interact directly at each layer – at the behavioral level, the robots create distributed control loops; at the executive level, they synchronize task execution; at the planning level, they use the TraderBots market-based techniques to assign tasks, form teams, and allocate resources.

Results
The market-based planning layer of each robot has two main components: a "trader" that participates in the market, auctioning and bidding on tasks, and a "scheduler" that determines task feasibility and cost for the trader, and interacts with the executive layer for task execution. This paper focuses on the trader

3 Please see Goldberg et al. [5], and Simmons et al. [9] for more details about this layered architecture.
component of the planning layer. In addition to the market component (or trader) of each robot, called the RoboTrader, the market of our architecture also contains OpTraders, traders that are similar in function except that: they are not associated with a robot, they provide a user interface to the system, and they act on behalf of the operators/users. When a user introduces a task into the system, it is the OpTrader that decomposes the task into its components, if necessary, and initially auctions off the task(s) while trying to minimize cost.

Our Mars exploration scenario is premised on the notion of scientific return, i.e., that a group of robots would be sent to Mars for the (potentially) valuable information they gather and return to Earth. We envision a scenario where a colony of heterogeneous robots is deployed on Mars. Scientists on Earth communicate high-level task descriptions to the colony (e.g., “find and gather data on several carbonate rocks”). We assume that communications limitations (bandwidth, delays, blackouts) necessitate highly autonomous robots, and preclude effective tele-operation of the robots or micro-managing of task execution by the scientists. The robots are therefore responsible for deciding which/how tasks are to be accomplished based on, among other things, the tasks' relative priorities. The goal for the robots is to utilize their time, resources, and capabilities efficiently so as to provide the highest possible scientific return on the tasks they are given.

In terms of the development and testing of our current system, we have focused on a characterize task that will fit within the broader scenario. In this task, a user/scientist specifies a region on the Mars surface, indicating that rocks within that region are to be characterized with an appropriate sensing instrument. The scientist may also specify the locations of rocks, if known. With respect to testing, a 3D graphical simulator developed for the project currently provides the “physical” robots and environment required (Figure 2), though, in the future, we hope to use real robots as well.

**Experiments**

The goal of the experiments presented here was to examine the effects of some of the major parameters of our market-based planning layer, in particular, those that impact participation in RoboTrader auctions.

These experiments, representing some of the early empirical results of our system, use a fairly simple experimental scenario. All of the experiments use the same initial setup: **6 rovers**, each possessing a RoboTrader, start clustered near the center of a rock field; **1 OpTrader** initially allocates (auctions) **50 characterization tasks**, each one for a specific, individual rock with a known location. The RoboTraders may also conduct their own auctions, buying and selling tasks amongst themselves to try to achieve a more efficient allocation/solution.

Efficiency here is based on cost minimization, where cost is calculated as the distance traveled by the rovers (in meters), plus an additional fixed cost of 10 (meters) for each rock characterized.

In the experiments presented here, execution is delayed until market quiescence. In other words, when every RoboTrader that wants to (or can) participate in auctions has done so for every auction available, and no tasks have been bought or sold, then the market is deemed to have quiesced. This means that a stable allocation (or local minimum) has been found. Only after quiescence do the RoboTraders execute their tasks. Delayed execution is useful in that it allows us to study the performance of the market under different parameters without the confounding effect of having tasks executing, and thus disappearing from the market.

The experimental parameters we examined form a progression in terms of participation in RoboTrader auctions. At one extreme, the RoboTraders are prevented from conducting auctions, meaning that there is no participation in RoboTrader auctions, and the initial OpTrader allocation constitutes the entire solution. In our current system, a bidder may send multiple bids to a particular auction, but the auctioneer is restricted to accept at most one of those bids. This allows the bidder to estimate the cost of its bids independently. Auction clearing is currently performed in a greedy manner, with the least cost bid (for OpTrader auctioneers), or highest savings bid (for RoboTraders auctioneers), being awarded first.
In addition, in the current system, only individual (rock characterization) tasks are traded; no bundled bids or awards are used. These restrictions (i.e., at most one award per bidder, greedy clearing, and non-bundled bids) imply that that the number of awards made per auction may impact the quality of the solution. For example, an optimal solution may involve awarding a particular bidder two tasks from the same auction, but this may only happen if one award is made per auction over multiple auctions. When the OpTrader allocation represents the full solution, a careful initial allocation is critical. At the other extreme is the case where full participation is allowed in RoboTrader auctions, i.e., every RoboTrader is able to participate in all of the auctions of every other RoboTrader. One drawback of such a situation is that the accuracy of the costs associated with RoboTrader bids cannot be guaranteed. As an example, consider the case of a RoboTrader bidder participating in two auctions occurring simultaneously. The RoboTrader assumes that its bid costs are independent for the two auctions, but this may only be true if it receives an award from at most one of the auctions. If the RoboTrader receives an award from both auctions, the false assumption of auction independence could lead to an inefficient allocation. Fortunately, this inefficiency may partially be rectified in later RoboTrader auctions.

In between both extremes is the case where RoboTrader auctions are allowed, and bid costs between auctions are ensured to be independent. This independence is achieved via two mechanisms. First, RoboTraders are prevented from participating in more than one external auction at a time. This serialization of participation ensures the independence of bids. Second, RoboTraders always set their own auction deadlines to expire after any external auctions in which they are participating as bidders. This means that when it comes time to clear its own auction, the auctioneer has no outstanding bids in other auctions that may alter its costs.

In order to examine these different levels of participation, as well as the initial allocation quality, we used the following progression of system parameters:

- **No RoboTrader Auctions (RTno):** RoboTraders never auction their tasks, so the OpTrader allocation is never altered and is the final solution. Participation in RoboTrader auctions is nonexistent.
- **OpTrader Auctions First (OTfirst):** the OpTrader auctions all of its tasks first, and only when it is finished do the RoboTraders conduct their own auctions. Participation in RoboTrader auctions is limited to occur only after OpTrader auctions.
- **Constrained Trader Auctions (Tcons):** the OpTrader and RoboTraders conduct auctions simultaneously, but RoboTrader bid costs are independent. This is achieved by the following constraints: a RoboTrader may participate as a bidder in at most one external auction, and that auction must terminate before its own auction is cleared. Participation in RoboTrader auctions may occur during the entire trial, but it is not full participation, being limited by conflicts arising from the constraints.
- **Round Robin Auctions (Trobin):** the OpTrader and RoboTraders participate in a round robin series of auctions, where each trader has the exclusive right to hold an auction during its turn. While not necessarily efficient or realistic, the round robin auction-synchronization mechanism enables full participation in RoboTrader auctions while maintaining independent costing.
- **Relaxed Participation (RTrelax):** each RoboTrader is allowed to participate as a bidder in all external auctions, regardless of how many there are and how they overlap with its own auction. Similar to Trobin, this allows full participation in RoboTrader auctions, but with the possible drawback of inaccurate costing.
- **Maximum Awards Per Auction (MAPA):** the maximum number of tasks awarded by the OpTrader in a single auction, influencing the quality of the initial allocation.

For each of the MAPA values of 1, 3, and 6, we performed experiments with each of the five other parameters, giving 15 experimental combinations. We ran 5 trials of each combination, with the completion criterion that all 50 rocks be characterized. The experimental data collected for each trial included: the final total cost of the solution, the degree of participation in RoboTrader auctions, the number of RoboTrader trades made, and the time to market quiescence (Table 1 through Table 4). In all experiments, task execution was delayed until auction quiescence.

As a baseline for comparison with our cost values, we ran experiments to find an optimal solution to our problem using a genetic algorithms (GA) approach. The best total cost achieved was 777.

**Discussion**

There are a number of general trends in the data that are notable in the cost data of Table 1, Table 2, and Table 3. One trend is that the quality of the solution degrades (i.e., total cost increases) as the MAPA value increases. The reasons for this are fairly straightforward. In each auction in our system, the
OpTrader never awards more than a single task to each RoboTrader bidder, possibly producing inefficiencies. For example, if the OpTrader awards three tasks in a particular auction (MAPA \( \geq 3 \)), they are to three different RoboTraders, when a better allocation might have had two of the tasks going to the same bidder. Thus, smaller MAPA values tend to lead to better initial allocations, though more OpTrader auctions must be held, and hence, the solution becomes more time-intensive.

A good solution, given the current limitations of our system, relies significantly on the quality of the initial allocation made by the OpTrader (i.e., with a low MAPA value). When MAPA=1, the allocations provided by all of the variations are essentially equally good. When the initial OpTrader allocation degrades (at MAPA values of 3 and 6), having RoboTrader auctions tends to improve the solution, as is evident from comparing RTno to OTfirst, Tcons, Trobin, and RTrelax.

### Table 1: Mean total cost of the solution. Standard deviations are shown in parentheses.

<table>
<thead>
<tr>
<th>MAPA</th>
<th>6</th>
<th>3</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTno</td>
<td>967</td>
<td>836</td>
<td>795</td>
</tr>
<tr>
<td>(6.1)</td>
<td></td>
<td>(26.6)</td>
<td>(1.7)</td>
</tr>
<tr>
<td>OTfirst</td>
<td>920</td>
<td>809</td>
<td>795</td>
</tr>
<tr>
<td>(15.7)</td>
<td></td>
<td>(7.2)</td>
<td>(1.5)</td>
</tr>
<tr>
<td>Tcons</td>
<td>921</td>
<td>837</td>
<td>795</td>
</tr>
<tr>
<td>(17.3)</td>
<td></td>
<td>(40.3)</td>
<td>(4.4)</td>
</tr>
<tr>
<td>Trobin</td>
<td>826</td>
<td>798</td>
<td>798</td>
</tr>
<tr>
<td>(19.9)</td>
<td></td>
<td>(5.3)</td>
<td>(4.4)</td>
</tr>
<tr>
<td>RTrelax</td>
<td>875</td>
<td>800</td>
<td>794</td>
</tr>
<tr>
<td>(24.8)</td>
<td></td>
<td>(9.9)</td>
<td>(2.0)</td>
</tr>
</tbody>
</table>

### Table 3: Mean number of tasks sold between RoboTraders. Standard deviations are shown in parentheses.

<table>
<thead>
<tr>
<th>MAPA</th>
<th>6</th>
<th>3</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTno</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(0)</td>
<td></td>
<td>(0)</td>
<td>(0)</td>
</tr>
<tr>
<td>OTfirst</td>
<td>3.2</td>
<td>1.6</td>
<td>0</td>
</tr>
<tr>
<td>(1.5)</td>
<td>(0.9)</td>
<td>(0)</td>
<td></td>
</tr>
<tr>
<td>Tcons</td>
<td>4.6</td>
<td>2.0</td>
<td>1.4</td>
</tr>
<tr>
<td>(2.9)</td>
<td>(1.0)</td>
<td>(1.1)</td>
<td></td>
</tr>
<tr>
<td>Trobin</td>
<td>14.6</td>
<td>4.0</td>
<td>0</td>
</tr>
<tr>
<td>(1.1)</td>
<td>(0.7)</td>
<td>(0)</td>
<td></td>
</tr>
<tr>
<td>RTrelax</td>
<td>23.2</td>
<td>7.0</td>
<td>0.4</td>
</tr>
<tr>
<td>(1.8)</td>
<td>(4.6)</td>
<td>(0.5)</td>
<td></td>
</tr>
</tbody>
</table>

### Table 4: Mean time (in seconds) for the market to reach quiescence, i.e., for convergence to a solution. Standard deviations are shown in parentheses.

<table>
<thead>
<tr>
<th>MAPA</th>
<th>6</th>
<th>3</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTno</td>
<td>215</td>
<td>421</td>
<td>1236</td>
</tr>
<tr>
<td>(10)</td>
<td>(7)</td>
<td>(31)</td>
<td></td>
</tr>
<tr>
<td>OTfirst</td>
<td>1000</td>
<td>1291</td>
<td>2050</td>
</tr>
<tr>
<td>(484)</td>
<td>(210)</td>
<td>(326)</td>
<td></td>
</tr>
<tr>
<td>Tcons</td>
<td>1149</td>
<td>1442</td>
<td>2013</td>
</tr>
<tr>
<td>(169)</td>
<td>(230)</td>
<td>(253)</td>
<td></td>
</tr>
<tr>
<td>Trobin</td>
<td>1155</td>
<td>1886</td>
<td>4890</td>
</tr>
<tr>
<td>(42)</td>
<td>(37)</td>
<td>(142)</td>
<td></td>
</tr>
<tr>
<td>RTrelax</td>
<td>283</td>
<td>480</td>
<td>1300</td>
</tr>
<tr>
<td>(18)</td>
<td>(9)</td>
<td>(32)</td>
<td></td>
</tr>
</tbody>
</table>

In addition to conducting RoboTrader auctions, Table 1, Table 2, and Table 3 show that increased participation in those auctions tends to improve the solution. One item of note is that even though Trobin and RTrelax show full participation, for MAPA=6 the cost of RTrelax is significantly greater (at a p-value of 0.01). This may be attributed to the inaccurate costing that can arise in the RTrelax case.
allocation by the OpTrader as the MAPA value decreases. When the MAPA value is 1, the OpTrader must perform at least one auction per task (very slow). Comparing this to Table 1, we note that an excellent solution may be obtained in less time, using MAPA=6 and Trobin, or MAPA=3 and RTrelax. This underscores the potential benefits of improved participation.

**Conclusion and Future Work**

This paper presents a market-based multirobot coordination architecture ideally suited for the space application domain. Work to date includes detailed conceptual formulation and proof of concept in simulation and on a physical robot team. Future work will focus on further development and evaluation of the efficacy of the TraderBot approach for the implementation scenario described above. The goal of this work is to produce an efficient and robust market-based multirobot coordination architecture.

**Acknowledgements**

This research was sponsored in part by NASA, under contract "Heterogeneous Multi-Rover Coordination for Planetary Exploration" (contract number NCC2-1243). This research was also sponsored in part by DARPA, under contract "Cognitive Colonies" (contract number N66001-99-1-8921, monitored by SPAWAR). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies or endorsements, either expressed or implied, of the U.S. Government. The authors thank the members of the Cognitive Colonies group: Scott Thayer, Bruce Digney, Martial Hebert, Robert Zlot, Aaron Morris, Charles Smart, and Bart Nabbe and wish to acknowledge the contributions of the FIRE project team: Reid Simmons, Trey Smith, Steve Smith, Jeff Schneider, Drew Bagnell, Vincent Cicirello, David Apfelbaum who programmed and found the GA solution, and Stuart Anderson.

**References**


