Hoplites:
A Market Framework for Complex Tight Coordination in Multi-Agent Teams

Nidhi Kalra Tony Stentz Dave Ferguson

CMU-RI-TR-04-41

August 2004

Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Abstract

In this paper we present a new class of tasks for multi-robot teams: those that require constant complex interaction between teammates. Much research has been done in the area of multi-robot coordination, but no existing framework meets the technical demands of such tasks. We have developed Hoplites in response to the need for a more capable framework. Hoplites is a market-based framework that couples planning with both passive and active coordination strategies. It enables robots to change coordination strategies as the needs of the task change. Further, it efficiently facilitates tight coordination between multiple robots. We compare the performances of Hoplites and existing coordination frameworks in a security sweep domain. Our results show that Hoplites significantly improves the quality of solutions found by the team, particularly in the most complex instances of the domain.

Contents

1 Introduction

Like humans, robots can accomplish more in teams than individually. Teams can often complete the same tasks faster and more robustly than a single robot. More importantly, however, they can accomplish a plethora of tasks that simply cannot be done by a single robot. For instance, we could use a robot team to perform a security sweep of an area suspected to contain mobile adversaries. We would want the team to determine and execute quickly those paths through the environment that maximize the likelihood of detecting all adversaries. Or, we could employ the team to explore a hazardous environment on Mars with the restriction that no teammate may be stranded without communication access to the base station. Instead, perhaps we want robots to monitor an art gallery and ensure that certain exhibit wings are continuously monitored while others are periodically checked for intruders.

These three tasks have common defining features. Firstly, they are principally defined by constraints between robots. Suppose that in the Mars exploration mission, a communication relay can only exist between two robots if there is a direct line of sight between them. So, a robot may not be able to explore a particular area until it ensures that at least one teammate will remain in its line of sight.

Secondly, these constraints are defined implicitly in terms of their effects rather than explicitly. An instance of the security sweep problem might require that, in an effort to detect adversaries, there may be at most one solid obstacle between every pair of adjacent robots. Although there may be infinitely many positions that satisfy this constraint, finding these positions can require significant computation.

Thirdly, when making their initial decisions, robots may not have complete information about the environment, their own states, or their teammates' states. As robots accumulate new information during execution, they may find their earlier choices inadequate or suboptimal and replan for improved alternatives. For example, a robot providing a communication relay to a teammate may receive news of large obstacles in the environment, realize that it cannot maintain the relay as planned, and try to find a new course of action.

Fourthly, a robot may need to satisfy constraints with several teammates at the same time. In order to better distribute resources while monitoring a gallery, one robot could guard the intersection of two exhibit wings, allowing two of its teammates to branch out and guard adjacent wings. This central robot would need to work with both teammates at the same time.

We associate with these features specific technical requirements for the coordination mechanism. First, recall that a robot's actions can be severely constrained by its teammates' actions. To choose actions wisely, the robot must be able to evaluate the actual contributions of its own actions based on the simultaneous actions of its teammates. Moreover, because of uncertainty, this evaluation must occur frequently throughout execution to take into account new information and adjust actions accordingly. We call this *tight coordination* and discuss it in detail in the sections that follow.

Additionally, because of the complexity of the constraints between robots, extensive planning may be required to find configurations that satisfy them. Thus, the mechanism must facilitate long-term, *planned* tight coordination. Finally, since robots coordinate with multiple teammates, we may have a scenario where two robots A and

D are not directly interacting with each other, but A may be coordinating with B, B with C, and C with D. So, the actions of A may indirectly constrain the actions of D through B and C. The coordination mechanism must be able to plan around a chain of complex constraints in a computationally feasible manner.

The field of multi-robot coordination has received much attention from researchers in recent years, but only a small portion of the work in this field is in the areas of planned and tight coordination. To our knowledge, there is no existing system that meets the requirements we outlined and therefore no system can successfully accomplish the tasks we described. We have developed a new market-based framework that facilitates the sophisticated coordination required by these tasks. We have named our framework "Hoplites" after the ancient Greek infantrymen who specialized in tightly-coordinated maneuvers.

In this paper we present Hoplites. We begin in Section 2 with an outline of the different types of tasks for which we would employ a group of robots and then examine their different technical requirements. In Section 3, we motivate our work further via a detailed evaluation of the existing body of research on moderately and tightly coordinated robot teams. We explain the principles and strategies behind Hoplites in Section 4. In Section 5, we then describe a domain that requires a tightly-coordinated team and illustrate the application of Hoplites to this domain. We present and discuss our results in Section 6 and conclude in Section 7.

2 Tasks for Multi-Robot Teams

There is a multitude of tasks for which we would use a group of robots instead of an individual robot. These tasks can be placed along a spectrum based on why, in each case, a group is preferable. At one end of the spectrum, we would use a group purely for speed and robustness. At the other end, the task simply may not be achievable by an individual robot. In Section 2.1, we present these differences intuitively; in Section 2.2 we explain the corresponding differences in terms of coordination requirements.

2.1 The Spectrum of Tasks

First, consider tasks that an individual robot could accomplish alone, such as mapping a large building. Usually, these tasks are easily decomposable into independent subtasks such as mapping the first floor and then the second floor of the building. A group of robots can be very effective: the subtasks are divvied up among the team members which then complete their subtasks independently. By sheer numbers, the team is faster and more fault tolerant than the single robot.

In the middle, there are tasks that cannot be completed by a single robot because the task has several subtask components that need different robotic functionalities. For instance, the cleanup of a hazardous environment could require that each area must be checked for explosive gases at most ten minutes before it is cleaned. We could make a jack-of-all-trades robot that could perform both activities, making it a task that can be completed by one robot. However, for simplicity of design (as well as speed and robustness), we might prefer to employ a heterogeneous team of robots in which

some robots can clean and some robots can check for explosive gases. As before, the team can decompose the task into subtasks (i.e. "clean-room-A" and "check-gases-in-room-A") and allocate them to appropriate members. However, before completing its subtask, a robot may need to consider the timing constraints imposed on it by other robots completing their subtasks.

At the other end of the spectrum, there are those tasks that need significantly more interaction between team members. Consider a group of five robots carrying a heavy box. Firstly, it is nearly impossible to split this task into independent subtasks (we cannot split the box into pieces and distribute them to the team members). Secondly, even creating subtasks such as "carry-left-side-into-the-next-room" and "carry-right-side-into-the-next-room" that are constrained to start at the same time are unlikely to be successful: the robots could attempt different paths to the next room and so drop the box. These tasks are characterized by a high degree of interdependence between the effects of robots' actions. That is, the effect of an action taken by one robot greatly depends upon the simultaneous actions taken by its teammates. Consequently, these systems are particularly vulnerable to uncertainty in robots' states and imperfect execution.

2.2 Task Requirements

For multi-robot researchers, the key difference between these example tasks is the level of the coordination between team members that is necessary for successful completion. At one end of the spectrum, the coordination in teams performing tasks such as mapping and exploration is typically limited to task decomposition and allocation. Here, the capabilities and states of several robots may need to be considered simultaneously in order to make appropriate choices. However, once a robot has a subtask, it does not need to interact with its team members during task execution. A *loosely coordinated* team can satisfactorily complete these tasks. The vast majority of the research into coordination strategies for multi-robot systems is for loosely coordinated teams for problems such as mapping and exploration [1, 2], reconnaissance [3], emergency handling [4], and tracking [5]. Research is aimed at developing decomposition and allocation strategies that most effectively make use of the team's resources.

Then there are tasks in the middle of the spectrum such as our example of cleaning a hazardous environment. Such tasks can be divided into subtasks that can be completed by single robots, but these subtasks have timing constraints between them. In addition to coordinating to decompose and allocate the subtasks, robots need to coordinate to meet the timing constraints. However, once a robot is cleared to begin executing a particular subtask, it does not need to coordinate with its teammates further. Such teams of robots are called *moderately coordinated*. Much of the work on moderately coordinated teams deals with how constraints should be set, communicated, and then met by individual robots. Such teams have been used for box moving [6], surveillance and monitoring [7], and hospital maintenance [8].

Sometimes, it is difficult, if not impossible, for a task to be broken down into subtasks that can be independently executed by individual robots. Again, it is non-obvious how robots moving a box from location A to location B can split the job up if no single robot can move the box alone. At best, the task can be divided into smaller tempo-

ral segments (i.e. moving the box from A to C and from C to B), but these must still be simultaneously completed by more than one robot. It is not even enough for robots to develop an initial coordinated plan and then begin synchronized execution of that plan. Uncertainty in actuation, sensing, and the environment all but ensures that something will go awry, sometimes with severe consequences. To be truly effective, a team must be *tightly coordinated*: members must work closely together during both planning and execution to avoid violating task constraints. The body of research on tightly coordinated teams is relatively small; primarily it deals with complex manipulation [9, 10, 11], formations [12], and surveillance and monitoring [13, 14]. This work is aimed at developing close coordination techniques that are responsive to change and computationally feasible.

3 Related Work

In virtually all robotics problems, we would like to generate solutions that are as close to optimal as possible while still being computationally tractable. Unfortunately, these objectives are particularly in conflict with one another in the domain of multi-robot systems where the complexity of algorithms for computing optimal solutions can be exponential in the number of robots. In the general multi-agent literature, a range of approaches have emerged to negotiate this conflict. Virtually all of these approaches have been developed in the context of loosely-coordinated teams; here, we pay particular attention to the work that does deal with more tightly-coordinated teams. Specifically, we look at how well these systems facilitate tight coordination, planned coordination, and coordination with multiple teammates.

3.1 Centralized Approaches

At one end, centralized approaches employ a single agent to plan for the entire team. Theoretically, they can provide optimal solutions since the planner can simultaneously consider the entire environment and the interactions of all team members. In reality, centrally coordinating more than a few robots quickly becomes intractable since the complexity of algorithms that plan joint actions is usually exponential in the number of robots. Furthermore, these systems are slow to incorporate new environmental information since new information must be sent back to the planner which recomputes the entire team's plan, usually at significant computational expense. Consequently, it becomes difficult for robots to coordinate in real environments that are wrought with uncertainty. Finally, these systems also rely heavily on the planning agent and the communication network and if either fail, the team fails.

Centralized approaches have been used primarily in loosely coordinated systems for task allocation. Caloud et al. [15] centrally allocate a variety of tasks such as "go-to-location" and "retrieve-object" to robots that independently complete their tasks. Brummit and Stentz [16] use a centralized system to demonstrate the benefits of real-locating goal points for a group of robots visiting a set of locations.

Truly centralized systems are rarely used for tightly coordinating a group of robots. Khatib et al. [10] use complex control models to treat two n-d.o.f. robots as one

$2n$-d.o.f. robot performing a complex manipulation task. Their approach uses inverse kinematics to compute actions in terms of end-effector positions. Unfortunately, it becomes exceedingly complex as we increase the number of robots. Furthermore, it is unclear how robots can reorganize to form different groups as the task's needs change.

3.2 Distributed Approaches

At the other end of the spectrum, distributed approaches dominate current multi-robot research. In these approaches, each robot has knowledge of its own state and its immediate environment. Each retains local control and chooses its own actions. Distributed approaches can be differentiated further by the mechanism through which robots coordinate. A team may rely on an *emergent* strategy where the coordination is a byproduct of robots following a set of carefully crafted operational rules or behaviors. Alternatively, robots may intentionally coordinate via specific protocols.

3.2.1 Emergent Approaches

Robots may be able to achieve some level of coordination simply by following a specific set of rules in a particular environment. Coordination is a byproduct of the interaction of these rules. In *reactive* approaches, these rules follow a sense-act cycle rather than the canonical sense-plan-act cycle and can work well when the role of each robot is clearly specified. Brown and Jennings [17] employ two robots to push a box from one location to another. One robot steers the box along a path while the second robot pushes the box along the direction of its orientation. These robots tightly coordinate via this fast reactive loop. As demonstrated by this work, robots can successfully complete the task without explicitly coordinating or maintaining a complex internal state. However, by being limited to simple rules, reactive approaches are inadequate in even slightly more complex environments where robots may need to switch tasks or roles.

Behavior-based approaches are better equipped to handle these situations. Robots maintain an internal state and have a set of behaviors such as "visit-location" or "follow-teammate" that are tailored to the domain. Robots choose behaviors according to the current state of the team and the environment. They can work on different tasks as the environment or global task requirements change. They can also contribute to several tasks at the same time by executing multiple behaviors in parallel. The primary challenge for researchers of behavior-based approaches is to develop effective behavior selection strategies that allow robots to accomplish complex tasks without complex planning.

Both reactive and behavior-based strategies favor simplicity and speedy computation over optimality. Robots are able to coordinate tightly throughout execution. It may even be possible for a robot to coordinate with multiple teammates at the same time. However, coordination between robots is limited to relationships that can be expressed by simple conditional rules, making them insufficient in domains that require planning. Matarić [18] provides a more detailed discussion of the principles behind reactive and behavior-based robotic systems.

Behavior based approaches have been used primarily in loosely coordinated teams engaged in exploration and mapping [2], emergency handling [4], and object track-

ing [5]. Less often, they are used to tightly coordinate teams tasked with moving in formation or exploring under communication constraints; we discuss this research in detail.

Much work has been done on formation control for multi-robot teams; Bahceci et al. [19] provide a comprehensive review of this work. Behavior-based approaches are particularly popular for maintaining robot formations because the constraints between the robots are simple and explicitly defined [20, 21, 22, 23]. For brevity, we use the work by Balch and Hybinette [23] as an illustrative example. In this work, behaviors take the form of potential fields. Each robot has a set of predetermined "snap-to" locations to which other robots are attracted. Robots also have motivations to avoid obstacles and move towards the goal. By combining these potentials, robots independently determine their position in the formation and maintain it well even while moving in the presence of obstacles.

Nguyen et al. [13] use a behavior based system to provide communication between a teleoperated robot and a base station via mobile relay nodes. Each node follows the node ahead of it until the strength of the communication link with the node behind it falls below some threshold. Then, it stops moving and drops out of line. This task requires fairly tight coordination between robots: a node must have up-to-date information about the node in front in order to follow it and up-to-date information about the link behind in order to decide when to stop. If a node follows when it ought to have stopped, the communication link between the base station and the robot can break and cause a mission failure or even a loss of the robot.

Although the system developed by Nguyen et al. successfully facilitates coordination with simple behaviors, it is difficult to generalize this success to other applications because the behaviors are very domain specific. Furthermore, it is unclear how well this system would perform if we extended the domain slightly to include multiple teleoperated robots that share the group of mobile relay nodes. This is similar to the Mars exploration domain. We expect that simple behaviors without planning would trap the system in local minima. For instance, if a predetermined number of nodes followed each teleoperated robot, they might provide excessive relay facilities to one robot while leaving another unable to complete its task because of communication constraints. Without planning and explicit coordination, the team cannot make use of collective state information that might highlight relocation strategies and result in better node distributions.

Wagner and Arkin [14] use a behavior-based approach for a similar communication-sensitive reconnaissance task: robots explore areas while trying to maintain network connectivity with each other. Their philosophy behind the approach is that, although the details of a particular task may change from instance to instance, the general approach to solving that task remains the same. With this in mind, they precompute several general behaviors that are necessary to complete the task and also precompute decision trees that place emphasis on different behaviors depending on the state of the environment. The result is "advice" that is given to a reactive controller. They achieve coordination between robots by preventing transitions from one task to another until all robots working on the current task are finished. If it is required by the overall task, a *plan controller* can also assign different roles to different robots.

Wagner and Arkin compare their work to behavior-based systems that do not use

this collection of precompiled behaviors or the coordination strategy. While the results appear promising in this domain, the system suffers from one major drawback: the human operator must decompose each task by hand into its components and then determine the relative importance of each component under all possible environmental states. This implies that the system will not scale well to many or more complex tasks. Also, it is unclear how much of the success can be attributed to fine-tuning the parameters and how much can be attributed to the principles behind the work. Lastly, note that the robots in this system are less tightly coordinated than the system developed by Nguyen et al. [13] as there is no tight loop between different members and the actions of one robot have relatively less effect on the actions of its teammates.

3.2.2 Intentional Approaches

Much work has also been done to develop strategies for *intentional* coordination in which robots communicate with each other to coordinate their efforts explicitly. By intentionally coordinating, robots can interact in more complex ways than are possible in emergent systems. However, to our knowledge, the benefits of intentional coordination strategies have only been harnessed to decompose and allocate tasks and not to tightly coordinate robots. In work by Zlot et al. [1], robots use a *market-based approach* to explore an environment in a loosely-coordinated system. That is, robots model an economy in which tasks such as "explore-location-xy" are bought and sold over a market via auctions. The auctions provide a distributed, fault tolerant way for robots both to distribute and to subcontract tasks that can be completed asynchronously and independently by individual robots.

Similarly, Botelho and Alami [24, 8] present M+, a negotiation protocol for the allocation of tasks that can be completed by individual robots. Unlike in the previous work, these tasks are also partially ordered, requiring a moderately coordinated team. They achieve the additional coordination required to meet temporal constraints simply: robots broadcast the start and end of tasks, indicating implicitly when other tasks can be started. This work is demonstrated in simulation in the context of hospital maintenance tasks and load transfer tasks.

A market approach is also used by Gerkey and Matarić [6] to allocate object tracking, sentry duty, cleanup, and monitor tasks in a loosely-coupled system. They also address the problem of box pushing in which one "watcher" robot observes a box and serially holds auctions for tasks such as "push-right-side-of-box" and "push-left-side-of-box." Two other "pusher" robots are better equipped to move the box, and they bid on and complete these tasks.

We disagree with Gerkey and Matarić's claim that their approach to box pushing is tightly coordinated and take this opportunity to highlight just what is meant by *tight* coordination. First, each robot can complete tasks "push-left-side-of-box" and "push-right-side-of-box" independently. As their work demonstrates, one pusher and one watcher can complete the entire task, implying that tight coordination does not exist between the two pushers (as it would if they were lifting the box). Secondly, tight coordination does not exist between the pusher and the watcher either. Consider a scenario in which there are several boxes that need pushing, several pushers, and several watchers holding auctions for pushing tasks. If a pusher p_1 currently attending box

b_1 abruptly switches to attending box b_2, there will be no dire consequences for the watcher attending b_1; it may simply need to find a replacement pusher. (Again, this would not be the case if the robots were carrying the box). Indeed, this is one of the most promising features of such task allocation strategies: they facilitate the efficient and transparent exchange of tasks which results in a highly fault-tolerant system. In this case, however, it also means that the system is only moderately coordinated. It is most similar to the clean-up task mentioned in Section 1: each subtask (i.e. "push-right-side-of-box") can be completed by one robot but these subtasks are ordered (we must alternate pushing the left and right sides to move it effectively). This ordering is achieved implicitly as the watcher only auctions tasks that can be completed immediately.

Lemaire et al. [7] also use a market-based approach in the domain of constrained exploration using multiple UAVs. The tasks are temporally constrained in that "traverse-path-p" must occur within a certain time frame and may be ordered with respect to other tasks. As in the Mars exploration domain, they are further constrained by the requirement that UAVs must maintain a communication link with a central station via relays. So the parent exploration task may require that a child task "create-link-with-base-station" be completed simultaneously by another robot. First the parent task is auctioned over the market to a master robot. Once this task has been allocated, the master robot auctions the child task to another robot. The robots then form a temporary master-slave relationship in which the master robot establishes the temporal constraints of the child task for the slave robot.

The work by Lemaire et al. deals with much tighter task allocation problems and facilitates closer intentional coordination between robots than the previous systems. However, there are two simplifications that allow the problem to remain fundamentally one of task allocation and also make it easier than the Mars exploration domain. First, the framework requires that the mission be defined as a set of partially ordered discrete tasks. Consequently, exploration is limited to predetermined geometric paths. Second, there are no obstacles in the environment so it is simple to determine how a link can be maintained between two robots. Thus, there is no need for robots to develop complex configurations; the simplifications remove the complex planning requirement from the task.

3.2.3 Hybrid Approaches

As shown by Balch and Hybinette [23] and Nguyen et al. [13], emergent coordination strategies can be successful for tightly coordinating a group of robots. By mapping a few key state components to specific behaviors, robots can respond quickly and in a prescribed manner to the actions of their teammates. By the same token, however, these systems are rarely sufficient for elaborate tasks that may require planning and more complex relationships between robots.

In hopes of reaping the benefits of both types of coordination, some researchers have developed hybrid systems in which emergent coordination is incorporated into a larger framework of intentional coordination. That is, robots actively coordinate to determine their relative roles when performing a task that requires tight coordination, but actually accomplish the task using behavior-based or reactive approaches.

Chaimowicz et al. [9] employ a hybrid strategy to coordinate a group of robots carrying a box. One robot assumes the position of leader, plans a trajectory for the team and broadcasts its heading and velocity. In a behavior-based fashion, each follower robot uses this information and the information from its own sensors to set its own heading and velocity. The main contribution of this work is that robots intentionally coordinate to negotiate for and switch leader-follower roles as different robots become better suited to guide the team. However, the constraints between robots are still relatively simple.

In work by Naffin and Sukhatme [12], robots negotiate to organize and move in formation. Robots begin as singletons and, as they encounter each other, negotiate for leader-follower roles in the formation. The leader is chosen based on predetermined rules that dictate which robot has the positional advantage. The leader then negotiates for itself and its followers all future encounters with other robots. Using a reactive protocol, each follower simply maintains a certain distance and heading from its leader. In this bottom-up fashion, smaller clusters of robots can quickly join up to form larger clusters. When a join occurs, the resulting formation may not fit the desired overall structure. The top-most leader then gathers information from all formation members, centrally plans, and instructs individual robots to change positions. The negotiation layer is the main contribution of this research over existing work, specifically that of Balch and Hybinette [23]. Naffin and Sukhatme's approach appears to work well; unfortunately, there is no empirical comparison between the two approaches so it unclear how or even whether the negotiations actually improve the team's performance. Also, unlike in Balch and Hybinette's work, the robots operate in an obstacle-free environment. It would be important to know how the framework handles disruptions in motion. As in the framework developed by Chaimowicz et al. [9], the constraints between robots are simple and explicitly defined in the task description.

Simmons et al. [25] propose a framework for very sophisticated coordination between robots. Their work addresses the problem of a large group of heterogeneous robots (cranes, mobile cameras, and smaller manipulators) working together in large-scale construction. Ideally, an end user would begin by offering a complex construction task to the group that would be accepted by a robot that would act as foreman for the task. This foreman would then decompose the task, create a task tree with constraints between subtasks, and negotiate with other robots for their participation on different aspects of the task. Parts of the task tree would then be allocated to team members for actual execution and they, in turn, would coordinate directly with each other at different levels depending on the needs of the subtask. This approach uses a three-tiered scheme to allow direct coordination between the planning, executive, and behavioral levels of different robots. Negotiation for tasks and synchronization occurs at the planning and executive levels, respectively, and tight coordination is achieved through behavior-based loops between different robots.

Certainly the scenario depicted by Simmons et al. is a long-term goal for many researchers of multi-robot teams. Unfortunately, their actual implementation uses a fairly simplified version of this problem. A roving eye, a crane, and a smaller manipulator are tasked with docking a beam into two receptors. Firstly, a predetermined static plan for the task is given to the robots, circumventing the planning layer. Secondly, tight coordination does not actually occur between any two robots: robots make decisions based

purely on the current state of the environment and without considering their teammates' actions. For instance, the crane does not interact with the manipulator directly, nor does it explicitly consider the effects of the manipulator on the beam. Instead, it simply uses data collected from the roving eye to move the beam in a prescribed manner. While coordination obviously emerges, it does not occur in an intentional, dynamic way as the motivating scenario suggests.

Unfortunately, these simplifications remove what is perhaps the most interesting aspect of the coordination framework that Simmons et al. propose. That is a mechanism through which robots negotiate a strategy for completing a specific task, agree to the constraints, and set up behavior based loops to tightly coordinate and complete the task. Such a framework would allow robots to plan tasks with multiple teammates, monitor their own progress, and even detect and correct failures. This work is a necessary and successful step towards providing these features, but the end goal is still far from being achieved.

We have found no system in the literature of multi-robot coordination that meets the requirements of the constrained exploration, security sweeping, and art gallery tasks. We believe that Hoplites fills this gap in the research.

4 The Hoplites Framework

The difficulty of the art gallery, security sweep, and constrained exploration domains varies between instances of the domain and depends significantly on the complexity of the environment and the size of the team. A cluttered environment is more challenging than an obstacle-free environment that permits full network connectivity and visibility. Additionally, the same instance can be harder for a team of only five robots than for a team of fifty robots that has redundancy in both network service and visibility. The difficulty of the task can also vary within a single instance of the domain. Consider an example of the constrained exploration domain that contains only one tight cluster of obstacles in a corner. The empty portions of the environment can be easily explored, but the cluttered corner may be quite difficult to investigate. We illustrate these ideas further for the security sweep domain in Figure 2 in Section 5.1.

The philosophy behind Hoplites is that team members should be able to adapt their coordination strategy to the changing demands of the task. In simple situations, the team can complete the task faster if team members make decisions more locally and work together via a coordination mechanism that is light on both communication and computation. Indeed, this is the motivation for many emergent approaches to coordination. However, harder scenarios may require more complex interaction between teammates that, in turn, requires a more complex coordination mechanism.

Hoplites is a modified market framework that consists of two coordination mechanisms for easier and harder environments. We have dubbed these mechanisms "passive coordination" and "active coordination", respectively. When passively coordinating, robots quickly react to each other's actions and influence each other implicitly. When actively coordinating, robots try to influence each other's actions explicitly by buying and selling complex plans over a market.

We begin this section by reviewing market frameworks. In Sections 4.2 and 4.3,

we give more detailed descriptions of the coordination mechanisms. We then detail exactly how Hoplites meets the requirements we have outlined and lastly we delve into the complexities of the framework.

4.1 Market Frameworks

Market frameworks are based on the mechanisms developed by Smith [26] and were first explicitly proposed by Stentz and Dias [27] as efficient frameworks for allocating tasks to members of a robot team. Robots model a market economy in which individuals act out of self-interest. Each task a robot completes generates some revenue but also requires some cost expenditure. The revenue function, $R : T \rightarrow \Re$, is a mapping that reflects how the task affects the team's proximity to the overall mission goal. If the task benefits the team, the revenue is positive; if it is a setback for the team (as in the case of a constraint violation), the revenue is negative and acts as a penalty. The cost function, $C : R \rightarrow \Re^+$, is a mapping that reflects the quantity of resources the robot consumed, such as fuel or network bandwidth. Robots bid on tasks that will generate the greatest profit, defined as revenue minus cost. The idea is that, through auctions and negotiations, tasks will be efficiently awarded to the robots best able to complete them.

A market framework has several key advantages. Firstly, it has all the benefits of distributed approaches, including robustness, flexibility, and speed. These features have been demonstrated by a number of researchers [3, 1, 7, 6]. However, resources permitting, it can also be more centralized to produce better solutions [28]. A single robot can plan more cost-effective allocations for larger portions of the team, bid on the tasks, and use the cost savings to purchase the participation of team members. This creates an increase in profit for that robot and also means that the team completes the task more efficiently.

4.2 Passive Coordination

In Hoplites, robots also act in a self-interested manner. However, rather than selecting tasks that appear most profitable, they select action sequences or plans that appear most profitable. Robots begin by generating a set $P_{candidate}$ of plans that are possible in the environment. They evaluate the profitability of each plan $p_i \in P_{candidate}$ in the context of other teammates' actions and the environment using the function C a modified version of the functions R. The function $R : A \rightarrow \Re$ now maps *actions* to rewards, providing profit evaluation at a much finer granularity.

Once a robot has chosen its most profitable plan p_{max}, it broadcasts p_{max} to other teammates. The teammates use this information to reevaluate the expected profitability of their current plans, update the plans, and broadcast these changes back. In this way robots' planning cycles iteratively incorporate the most recent information about teammates' intentions. We hypothesize that this phase of Hoplites may suffer from some instability if robots' planning occurs at precisely the same time. They may simultaneously choose to change their plans and inadvertently violate task constraints. They may also oscillate back and forth between plans as they react in synchronization to each other's broadcast plans. We further hypothesize that, on real robots, there will

be enough uncertainty and external randomness that synchronization will be rare and brief. As we demonstrate in Section 6, the passive coordination phase alone can be effective in many environments where the correct actions are obvious to all parties.

4.3 Active Coordination

In complex environments, a passive strategy may generate suboptimal solutions because robots cannot change or guarantee their teammates' actions. To see this, first note that a robot's action choices may be affected by several teammates simultaneously and that the same robot can affect its teammates' actions in different ways. So, it can be difficult for one robot to predict how an action will affect its teammate without simultaneously considering the actions of all other teammates. Secondly, robots may change their plans at any time, so a robot cannot count on its teammates to actually follow through with their broadcast plans. Together, they suggest that in a passive strategy, robots cannot be very confident about their estimates of the profitability of particular actions.

Now, consider a scenario in which a particular plan is likely to reap some rewards for a robot no matter what its teammates are doing. We would call this a *low-risk* action. Suppose a different plan is *high-risk*. That is, under the right circumstances, it can be extremely profitable, but under the wrong circumstances, it can be quite unprofitable. Moreover, even if the current circumstances are right at one moment, the robot cannot be sure these conditions will last. Without a way to create and then guarantee the right circumstances, robots rightly shy away from high-risk plans in favor of safer ones, even though they may be less profitable and correspondingly suboptimal.

The active coordination mechanism steps in here via a market framework. When a robot discovers that its best plan is only marginally profitable, it tries to develop a team plan that consists of actions its teammates could take that would make its own plans more profitable. If it finds such a team plan, it requests price quotes from its teammates that indicate how much compensation they require before they will participate in the plan. If the robot stands to make more than it will spend on compensating its teammates (i.e. it has a positive profit margin), it makes a bid for its teammates' participation. If they refuse, then the robot adopts its original plan and continues with a passive coordination strategy until the next planning cycle. On the other hand, if they accept, the group is bound by a contract and every robot involved in the team plan must complete its portion. Once complete, the team members revert to a passive strategy and continue on their own. As with typical market frameworks, the more profitable team plan is overall a lower cost set of actions for the team and leads the team to complete the task more efficiently.

4.4 Hoplites and Task Requirements

In the previous sections we provided a very general overview of Hoplites. However, it is important to outline exactly how the system meets the requirements of tight coordination, planned coordination, and computationally feasible coordination between multiple robots. We detail each of these in turn.

4.4.1 Tight Coordination in Hoplites

We say that robot A coordinates with robot B if it considers the state of B when selecting its own actions. Further, we say that this coordination is *tight* if A regularly considers B's state throughout planning and execution.

In Hoplites there are two parameters, plan granularity and replanning frequency, that dictate just how tight the coordination is. By "plan granularity" we mean the interval of time between consecutive actions in the plan. A finer granularity corresponds to a shorter interval and implies that a robot samples its teammates' actions more frequently when choosing its own. The second factor is how often robots update their plans during execution. A higher replanning frequency means that a robot reevaluates its plan in the context of its neighbors actual activities more often. This allows it to react more quickly to unexpected changes in their actions. Both finer plans and a higher replanning frequency create tighter coordination.

4.4.2 Planned Coordination in Hoplites

By "planned coordination" we mean that a robot plans at some time t the interactions it will have with its teammates at a later time t'. The maximum difference $n = t' - t$ is known as the *planning horizon* and defines the length of a plan in terms of time. Planned coordination occurs in Hoplites in both the passive and active coordination phases. In the passive coordination phase, a robot creates its own plan in response to the plans its teammates have generated for themselves and subsequently broadcast. This allows a robot to anticipate the long term effects of its actions and works well when the environment is simple. In the active coordination phase, a robot intentionally plans out the interaction between itself and its teammates and tries to persuade its teammates to adopt this team plan. Here, robots can to some degree *determine* the effects of their actions. Such measures become necessary when the environment is complex.

In order to develop a team plan, a planner is required that can tractably search joint action spaces. The Hoplites framework does not specify any particular planner; the right choice depends upon the domain, the number of robots, and the importance of optimality. In our work we find that optimal planners such as A* and D* become computationally infeasible when search spaces are very large. Instead, we recommend and use search methods that sacrifice optimality for speed such as probabilistic roadmaps (PRMs) [29] or Rapidly Exploring Random Trees (RRTs) [30]. RRTs in particular have been shown to work well in high-dimensional spaces [31] and we use them in our own implementation to develop team plans in a computationally efficient manner.

4.4.3 Coordination Between Multiple Robots in Hoplites

Coordination between multiple robots occurs in many forms. Borrowing ideas from network and graph theory, we consider each coordination form in terms of its topology. We illustrate these ideas in the context of a team of five robots and depict some example scenarios in Figure 1, using solid bi-directional arrows to indicate passive coordination and dotted bi-directional arrows to indicate intentional coordination.

As depicted in Figure 1 *(a)*, robot A initially passively coordinates with teammates B, C, and D, and vice versa. (Although it is the case for most applications, passive co-

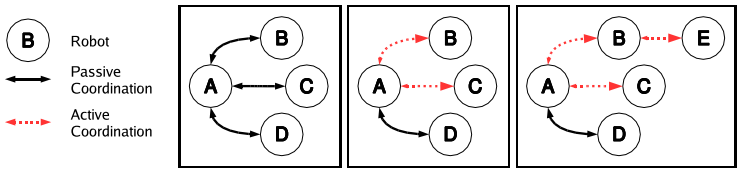


Figure 1: Illustrations of coordination topologies for a team of five robots. *(a)* A star topology in a passively coordinating group (A, B, C, and D). *(b)* A star topology in an actively coordinating group (A, B, and C). *(c)* A chain topology in an actively coordinating group (A, B, and E).

ordination is not necessarily bi-directional: it is conceivable that robot A is constrained by robot D but that D is not constrained by A). We can think of planning in this scenario as a navigation problem in which A must move from some start position (its current state) to some goal position (its desired future state). Each constraint placed on A by B, C, and D alters the terrain by making some portions of the space unnavigable (hard constraints) or more costly to navigate (soft constraints).

Here, the coordination resembles a star topology with A as the center node and B, C, and D as peripheral nodes. However, the structure of a passively coordinating team is very informal and constantly changes since any robot can passively coordinate with any teammate simply by altering its planning terrain to include constraints imposed by that teammate. Although an increasingly complex terrain (created by passive coordination with more teammates) may mean a longer search, the computational complexity of the search does not change. Thus, the complexity of planning during passive coordination is not greatly affected by topology.

Now, suppose A wishes to execute a particular plan by convincing both B and C to follow a team plan. This is depicted in Figure 1 *(b)*. The unequal sizes of the arrow heads indicates that the plan is directed from A to B and C. Although this active coordination has a star topology, it differs greatly from the same topology in a passively coordinating group. Rather than A planning for itself from one start state to one goal state in a complex terrain, it simultaneously plans for itself and B and C. This is similar to centralized approaches in that one agent leads several others. Consequently, it may suffer from some of the same drawbacks associated with centralized approaches such as computational intractability or slow responsiveness. Still, we can imagine how it may be a reasonable course of action when complex tight coordination is required between a small number of robots.

To employ such a topology, robot A first generates a team plan P_{ABC} (perhaps using a planner mentioned in Section 4.4.2). It then requests a price quote from both B and C, and, if it can afford *both* costs, it will send bids to them. Several features of Hoplites minimize the potential pitfalls of a star topology. Efficiency comes first from robots being able to independently and asynchronously investigate the potential benefits of a star topology in particular scenarios. So, unlike in centralized systems, the team does not come to a halt whenever planning occurs. Second, any centralized

planning is localized to the relevant subset of the team, making it much faster for the lead robot to generate a plan. Third, peripheral nodes B and C independently evaluate the team plan, enabling them to estimate better and faster the utility of the plan by incorporating local information that may not be available to the central planning agent A. Fourth, in many cases, B and C may be able to replan locally provided they do not violate constraints imposed by A, allowing the group to respond to changes in a simpler and faster manner than in a centralized system. We discuss replanning further in Section 4.5.3.

Now suppose that, unbeknownst to A, B is simultaneously coordinating passively with some robot E. Further suppose that the team plan P_{ABC} is a low-cost option for C but a high cost option for B because it violates the constraints imposed upon it by E. Then, it is likely that C's price quote will be low but B's quote may be too high to make the team plan viable for A. In an effort to increase its own profit, B may attempt to actively coordinate with E by developing a plan P_{BE} that makes P_{ABC} less costly. The plan components for B must be the same in both P_{ABC} and P_{BE}. If P_{BE} has a low cost, B can quote to A a lower price that includes E's cost for P_{BE} and B's new cost for P_{ABC}. As shown in Figure 1 *(c)*, the active coordination between A, B, and E has a chain topology. That is, A actively constrains B's actions and B, in turn, actively constrains E's actions.

Hoplites makes it possible to efficiently use a chain topology to serially coordinate a large number of robots. Firstly, planning for the chain occurs through a series of linked plans such as P_{ABC} and P_{BE} (and conceivably P_{EF}, P_{FG}, etc). So, at no time must a single agent plan for a large number of robots. Secondly, the utility of a plan is evaluated locally and compressed into a single figure, the price quote. By combining a series of price quotes, robots can concisely transmit the costs and benefits of a collection of plans. Thirdly, this makes it possible for robots to investigate different solutions to the same problem. The most cost effective one is naturally adopted in the market. Fourthly, if at some stage a link in the chain (suppose P_{BE}) must be broken or becomes too costly, it is possible to repair it locally (B could develop an alternative plan for itself and E or coordinate with another robot entirely). The result is that A need not be aware that P_{BE} exists, that E is making P_{ABC} possible, or that it is paying for E's participation in P_{BE}. Moreover, E need not be aware of why P_{BE} is profitable or that it is helping A in any way. Thus, the needs at one end of the team can be transmitted to and met by the other end of the team transparently and efficiently.

4.5 Framework Details

There are many issues that deserve further explanation, such as the method by which robots choose coordination strategies, generate team plans, and accommodate changes in their environment. We discuss each of these in detail in this section.

4.5.1 Choosing Coordination Strategies

In Hoplites, a robot passively coordinates until it expects that its best plan will have a decrease in profit, specifically due to a constraint violation. Unlike resource consumption, which is typically independent of other robots' actions, constraint violations

can often be avoided through intentional coordination. In practice, we have found that this criterion works well because robots choose to investigate team plans and actively coordinate only when there is a good chance that doing so will actually create better, more profitable solutions. In this way, the system is not bogged down by excessive computation.

We believe there is scope for robots learning when to coordinate instead of or in addition to using predetermined criteria. It is possible that robots may be able to identify other situations that could benefit from active coordination or particular types of constraint violations that do not require it.

4.5.2 Commitments

Commitments between teammates is an important and complex aspect of most market frameworks. We would like teammates to be committed enough to fulfill the agreements made with other robots, while still having the flexibility to abandon commitments if they turn out to be detrimental to the task.

We propose that the team plan agreed upon by two robots is binding until the end. The exception is when robots are able to negotiate breach-of-contract terms. Consider a scenario in which robots A and B agree to complete plan P_{AB}, but half-way through executing the plan, robot C bids on B's participation in some plan P_{BC}. If P_{AB} and P_{BC} are compatible and B can complete both simultaneously, it simply agrees to P_{BC}. On the other hand, perhaps the two are incompatible. Then, B requests from A the compensation it would require from B before it will release B from its commitments. This quote can include many factors such as the opportunity cost to A and the cost to A for finding a replacement for B. B adds this price into its price-quote to C. If C can still turn a profit, it pays B, which compensates A for the breach. In this way, the value of P_{AB} is explicitly weighed against the value of P_{BC}, and the more profitable of the two is preserved.

4.5.3 Replanning while Committed

In virtually all tasks, it is reasonable to expect that the state of the world will change. Robots can obtain new, more accurate information about their environment as well as about each other. It is desirable, then, for robots to incorporate this new information and update their plans to ensure that they are always taking the best actions given the information available and their computational abilities. This is particularly important when robots are engaged in team plans since the interaction between teammates can be significant and complex.

First, each robot must continuously reevaluate its portion of the team plan in the context of the environment and its teammates to determine whether it will be less profitable than originally expected. When a robot finds that the plan will be less profitable, its simplest option is to see if there is a more profitable plan that it can execute independently. If such a plan exists, it can negotiate a breach-of-contract price with its teammates as described in Section 4.5.2 and cut its losses by abandoning the team plan. On the other hand, it may attempt to modify its team plan to accommodate new information without abandoning its teammates. If it finds such a modification, it can request

from its teammates compensation for the unexpected costs incurred, and continue participating in the team plan. Indeed, a robot should investigate both avenues and choose the more profitable of the two.

4.5.4 Selling Plans vs. Selling Participation

We have suggested that a robot A can increase the profit of a particular plan p by developing a team plan P that better complements p and selling P to specific teammates. However, suppose that there are a number of different teammates that could take actions to make p more profitable. Instead of A investigating all possibilities, it may be more effective for A to send p to its teammates which generate plans individually and try to sell their participation to A. The differences between these strategies are a product of the differences between central and local planning. When coordination must occur between many teammates or is particularly complex, a single plan developed by A may be most effective. When coordination is less complex or when several teammates can provide the same service, a comparison of several locally-developed plans may be better. We intend to investigate empirically the appropriateness of each strategy on different domains.

5 Hoplites and the Security Sweep Domain

We have chosen to demonstrate Hoplites in a security sweep domain in which robots with a limited sensor range sweep a cluttered environment for mobile adversaries. The aim is to sweep the area as quickly as possible, moving from a start location to a goal location, while preventing adversaries from remaining in the area undetected at the end of the sweep. This task has obvious applications in the fields of security, surveillance, and reconnaissance. We begin by explaining how the security sweep domain has the same requirements we listed in Section 1 and then briefly discuss the prior work on this and similar problems. We then detail our own implementation, define revenue and cost functions used by the market mechanism, and illustrate how Hoplites is used in this domain.

5.1 Requirements

Firstly, this domain requires multiple robots. Consider the simple environment in Figure 2 *(a)* where robot A must sweep an area with one obstacle. Topologically, this environment is multiply connected. As we can see from the two trajectories in the figure, it is impossible for A to guarantee that it has seen all adversaries: the adversary can hide indefinitely in the blind spot created by the obstacle. At least one other robot is required to complete the task.

Secondly the domain requires tight coordination. In Figure 2 *(b)*, robots A and B must sweep a corridor with two obstacles. Although they only need to move in a straight line to clear the area, their timing must be right. If they remain in lock step, all adversaries will be detected. However, suppose A moves faster than B and at some time t they are at waypoints 1 and 2, respectively. Then, an adversary hiding at waypoint 3

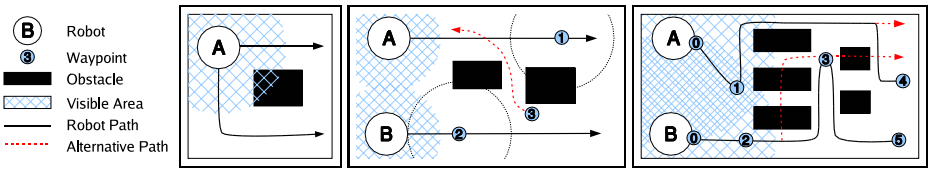


Figure 2: An illustration of the requirements of the security sweep domain. *(a)* A simple environment that cannot be swept by a single robot. *(b)* Although robots may be able to use a simple strategy to select optimal actions, they must tightly coordinate to time their trajectories properly. *(c)* In complex environments, the coordination must be planned in advance and constantly replanned to incorporate new information. When robots have shorter sensor ranges, at least three are required to sweep this area.

could follow the path marked in red and remain in the corridor without being seen by either robot. Even when their actions are simple, robots must work together closely to successfully complete the task.

Thirdly, in complex environments, the domain requires planned tight coordination. In Figure 2 *(c)*, A and B must sweep a corridor with two groups of obstacles. A successful strategy is for A to secure the already swept portions of the corridor while B moves ahead to sweep the next portion. We have plotted in black the trajectories for each robot with timing constraints at particular waypoints marked in blue circles. A robot at waypoint i cannot continue to its next waypoint k until all waypoints j where $i < j < k$ have been visited. So, B cannot begin moving to waypoint 2 until A has arrived at waypoint 1. When A arrives at waypoint 1, it must then wait for B to move to waypoints 2 and then 3 before it can continue to waypoint 4. Clearly, A and B must actively plan their coordination in order to complete the task; an emergent strategy would be insufficient in this environment.

If A and B had perfect information about the environment, they could perhaps plan once at the beginning, find this coordinated path, and execute it with minimal communication. However, in real environments, A and B have no prior information. From their starting points at waypoints 0, they can only see the the edges of the first three obstacles and cannot tell how large they are or whether there are more obstacles behind them. We mark in red how the initial paths generated with partial information diverge from the paths generated with perfect information. As the robots execute this imperfect path, they will find that the first three obstacles are larger than they initially appeared and have to adjust their paths to accommodate this change. When they observe the remaining two obstacles, they will have to drastically change their original plans. Moreover, the perception and actuation for real robots is not perfect so robots may have uncertain or even incorrect map or state information. Robots must constantly replan their coordination to incorporate the latest information about their own states and the environment.

Finally, by reducing the range of A and B's sensors to be less than the width of the corridor, the two cannot alone clear the environment in Figure 2 *(c)*. The environment

requires that the swept area be held secure at waypoint 1 while the unseen area is swept. With a reduced sensor range, at least two robots are required to secure this swept area. Thus, it is easy to see how this domain can require the coordination of several robots simultaneously.

5.2 Related Work

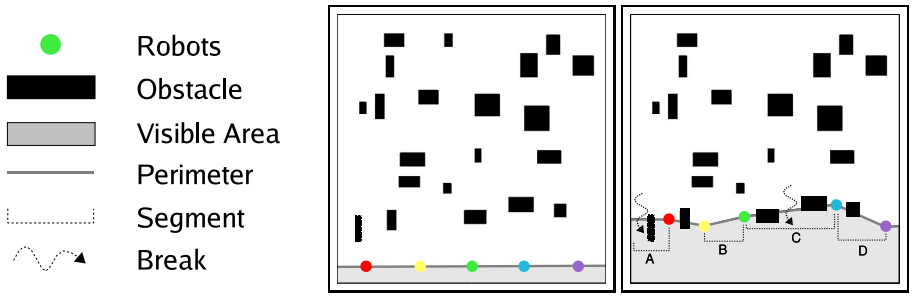
Our approach to the security sweep domain is loosely based on the set of plane sweep algorithms used for solving many problems in computational geometry. In these algorithms, a vertical line is dragged from left to right across a plane containing data. Information about the data is gathered when the line intersects that data. The idea is that, once the line has completely traversed the plane, all desired information about the environment should be known. Applications of sweep algorithms include the computation of Voronoi diagrams, map overlays, and nearest neighbors. Mehlhorn [32] provides a detailed description of the general algorithm and Van Kreveld [33] provides a thorough list of specific plane sweep applications and algorithms. In our domain, we sweep a plane containing obstacles with a line of robots. When the robots reach the other end of the plane, no adversaries should remain undetected.

The security sweep domain is related to the domain of pursuit evasion, in which one or more *pursuers* must find all the *evaders* that are moving in an environment [34, 35]. However, nearly all of the work on pursuit evasion is aimed at developing purely geometric algorithms for different instances of the domain. These geometric algorithms require several simplifications that make the domain much less rich than the security sweep domain. Researchers frequently assume a completely known environment [36, 37, 38, 39], permit only single searchers [40], or both [34, 35, 41, 42, 43].

The exception is work by Kim et al. [44] in which unmanned ground and air vehicles pursue an embodied evader in an unknown environment. The aerial vehicle scans the ground for the evader and, upon detection, broadcasts the evader's location to the pursuers. The pursuers then use local behaviors to move towards the evaders and catch it by coming within some distance of it. The primary effort of this work is to develop local one-step policies for the ground vehicles that result in the fastest capture. It does not address coordination between the ground vehicles.

5.3 Domain Details

Figure 3 *(a)* is a sample room of dimension (X_{max}, Y_{max}) in which there are twenty obstacles. The robots (marked as colored circles) begin in a line at the bottom of the room ($y = 0$) and have no prior knowledge of the number of obstacles or their locations. Their task is to sweep the area with their sensors as they move to the top of the room ($y = Y_{max}$). The line of sight between the robots forms the forward perimeter of the sweep. The area behind this line (shaded in light gray) has already been swept while the area ahead of it is unswept. Robots must move in a way that minimizes the blind spots along this line that would allow a foe to enter undetected into the already swept area. We refer to these blind spots as *breaks* in the perimeter. Robots must coordinate to avoid breaking the perimeter as they traverse highly cluttered environments.



(a) (b)

Figure 3: An illustration of the security sweep task. *(a)* A sample room to be swept. *(b)* Secure (B and D) and broken (A and C) perimeter segments.

We place two restrictions on this domain. These restrictions do not affect the technical requirements of the problem; instead they remove some of the domain-specific complexities that do not contribute to our goal of developing a general coordination framework. Firstly, we restrict the adversaries to the open spaces so they cannot hide inside obstacles. This is comparable to a team of mobile robots sweeping for vehicles among a cluster of buildings. Secondly, we assume that obstacles are rectangular and oriented along the x and y axes. This makes it easier for robots to detect and model obstacles. Complex obstacles such as arbitrary polygons make the sweeping aspects of the problem more complex but would not add to the coordination aspects.

Figure 3 *(b)* illustrates exactly what constitutes a perimeter break. The team's perimeter is composed of several segments defined by the straight lines connecting adjacent robots. The leftmost and rightmost segments join the first and last robots to the left and right walls of the room, respectively. In the figure we examine four segments marked A, B, C, and D. Segment A is broken because there is an obstacle between the leftmost robot and the left wall. An adversary could slip between the obstacle and the wall and into the swept area without being detected. Segment B is secure because the two robots have a clear line of sight to each other, so an adversary cannot sneak between them. However, segment C is broken because there are two obstacles between adjacent robots. There is a gap between the two obstacles that provides a stealthy path into the secured area. Finally, segment D is secure because, although the robots cannot see each other, the obstacle between them is solid and can form part of the perimeter.

5.4 Revenue and Cost

We identify one source of revenue in the security sweep domain: the progress across the environment. With n robots working in an environment of dimensions (X_{max}, Y_{max}), each robot is responsible for sweeping an imaginary corridor along they y axis of width X_{max}/n. Specifically, each step a robot takes in the y direction generates for it revenue proportional to X_{max}/n. If a robot moves backwards towards $y = 0$, it is penalized accordingly. This gives preference to the most direct route from $y = 0$ to $y = Y_{max}$.

There are two sources of resource consumption costs as well. First, robots are penalized proportional to the amount of time they take to cross the environment. This cost represents the value of completing the task in a timely manner and discourages robots from sitting idle. Secondly, robots are penalized proportional to the actual distance they travel. This cost represents the value of energy (battery life or fuel) and discourages robots from taking winding paths through the environment.

Perimeter breaks are the only source of penalties from constraint violations. A robot incurs a penalty for each step it takes that breaks a previously-secured perimeter segment or fails to secure a previously-broken perimeter segment. A robot is only responsible for keeping intact the two segments between it and its adjacent teammates (or, in the case of the first and last robots, the segments between them and the walls). The size of the penalty determines the importance of conducting a thorough sweep relative to the importance of conserving time and energy. So, a relatively high penalty encourages robots to expend both time and energy in an effort to take paths that keep the perimeter intact. A lower penalty encourages robots to traverse the environment quickly, permitting breaks when they are difficult to avoid. We might prefer the former in a security mission and the latter in a stealthy reconnaissance mission.

5.5 Illustration of the Framework

The Hoplites framework can dramatically improve the team's performance in the security sweep domain by facilitating intentional tight coordination between several robots. Suppose that we use a team of four robots to sweep areas and that the thoroughness of the sweep is of utmost importance (the penalty for a perimeter break is much higher than distance and time costs). Consider the room illustrated in Figure 4 *(a)*. In this simple environment containing only one obstacle, each robot initially selects a straight path (P_{*1}) as its best action, marked by a solid black arrow. When a robot receives notice of its teammates' intended trajectories, it still finds that its current forward path is most profitable. In one step, the team converges to the optimal plan using only passive coordination.

However, as shown in Figure 4 *(b)*, the addition of one obstacle makes this approach suboptimal. The local environments of B, C, and D have not changed, so they will still initially choose the same straight path as before. However, A must now choose between following the path P_{A2} marked in red or continuing with path P_{A1} between the two obstacles. At this stage, both options have the same penalties for A in terms of perimeter breaks. If A chooses P_{A1}, B will not face a penalty and will continue following P_{B1}. If A chooses P_{A2}, however, B will have to make a choice between P_{B1} and P_{B2}. Now, B will be forced to choose P_{B2} to keep the perimeter intact between itself and A and avoid a large penalty. Indeed, the optimal solution is for both A and B to follow the P_{A2} and P_{B2}, respectively.

Unfortunately, using a passive coordination strategy, A will choose P_{A1} over P_{A2}. Although P_{A1} and P_{A2} have the same number of breaks in the perimeter, P_{A2} has larger time and distance costs, so it is overall more costly than P_{A1}. Even if A knew that following P_{A2} might cause B to follow P_{B2}, it could not guarantee B's actions. So, A would choose the low-risk path P_{A1} over the high-risk path P_{A2}, resulting in a suboptimal solution for the team.

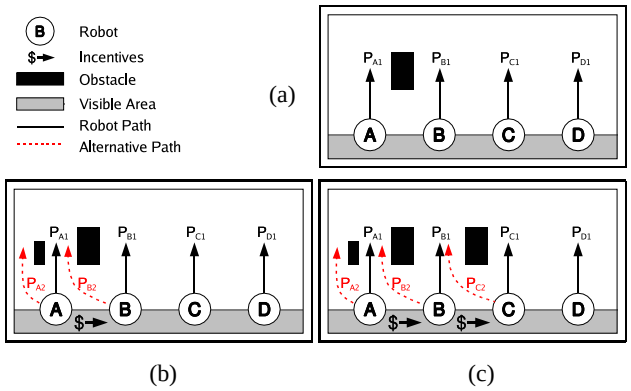


Figure 4: Simple examples motivating active coordination. *(a)* An environment where passive coordination suffices. *(b)* An environment requiring active coordination. *(c)* An environment requiring active coordination between several teammates.

Through the market, Hoplites facilitates intentional coordination between A and B, making P_{A2} a sure bet for A. During planning, A finds that it is guaranteed to incur a minimum cost of C_a due to penalties from perimeter breaks. Since this is a constraint violation, it searches for a team plan P_{AB} that would decrease this cost (clearly, $P_{AB} = \{P_{A2}, P_{B2}\}$). A then requests a price quote Q_b from B for P_{AB}. This quote reflects the additional cost C_b to B for taking P_{B2} instead of its default path P_{B1}. We expect that, since the penalty for perimeter breaks is higher than the costs of consuming resources, C_b will be less than C_a and A will have a profit margin $M = C_a - Q_b = C_a - C_b$. A can then offer Q_b to B to offset its cost and pocket M as its own additional profit. A could further offer part of M to B if B had competing alternatives to P_{AB}. Now, B will find that it can do at least as well as it would have with P_{B1}, so it will accept the offer. In the pursuit of larger profits via the market, the team arrives at the optimal solution.

Hoplites also enables the explicit coordination of several robots in a chain. In Figure 4 *(c)*, we add a third obstacle to the environment. Although this changes B and C's local environments, it does not create any breaks in their paths. Consequently, they will initially choose the same paths P_{*1} as in the previous environments. Also as before, only A faces a break in its perimeter. However, unlike in Figure 4 *(b)*, B's price quote Q_b for P_{AB} will be much higher than before because P_{B2} will cause breaks between B and C and cause C_b to be much higher. Indeed, these breaks will be much more costly than the breaks between A and the wall since the obstacle between B and C is much larger than the obstacle between A and the wall. So, C_b will be larger than C_a and A will be unable to make any offers to B.

However, B can generate the team plan $P_{BC} = \{P_{B2}, P_{C2}\}$ and request a quote Q_c from C that reflects the added cost C_c of taking P_{C2} instead of P_{C1}. We expect C_c to be small since P_{C2} only constitutes a small change in C's plan and does not create any perimeter breaks for C in the context of P_{BC}. Thus, upon receiving Q_c, B can evaluate its new cost C'_b that is the cost of taking P_{B2} instead of P_{B1} when C

simultaneously takes P_{C2}. It can then send to A a new price quote $Q'_b = C'_b + Q_c$. Again, we expect that $C'_b + C_c$ will be less than C_a, so A will be have a profit margin $M = C_i - Q'_b = C_i - (C'_b + C_c)$. A will then offer Q'_b to B which, in turn, will offer Q_c to C. Both B and C find that they can do at least as well as they would have with their original paths, so they will accept the respective offers.

Here, the system is able to find the more complex optimal solution via the market. Interestingly, neither A nor C is aware that the former is paying for the participation of the latter. This activity is invisible to everybody but B. In this manner, needs at one end of the sweep can be transmitted transparently to robots at the other end. This can be extended to an arbitrarily long chain of robots.

6 Experiments and Results

We have performed experiments in simulation comparing our implementation of Hoplites on the security sweep domain to three other frameworks. We have used the MVERT framework [5] developed by Stroupe as a baseline for comparison and incrementally improved it to create P-MVERT and PC-MVERT.

6.1 Frameworks

6.1.1 MVERT

MVERT [5] is a behavior-based framework that was originally used for foraging and tracking tasks for groups of robots. In MVERT, a robot cannot explicitly communicate with its teammates but can observe through its sensors their current locations. A robot first estimates the action every other team member will take next. Then, it chooses for itself the action that is most valuable given the expected contributions of the team in the next time step. Each robot repeats this procedure of selecting its next best action.

We believe that MVERT is the most appropriate approach to use for comparison for two reasons. Firstly, like other behavior-based approaches and unlike intention approaches, it can be applied to tasks that cannot be divided into subtasks. As Stroupe notes, "The potential applications of MVERT include any tasks that can be represented by some computable mathematical function" [5]. Secondly, in contrast to other behavior-based approaches, it allows robots to plan their coordination by anticipating future contributions of their teammates. Stroupe demonstrates that MVERT outperforms other approaches on several domains specifically because of this feature. Consequently, we believe that MVERT is the most competitive framework with Hoplites.

We have applied MVERT to the security sweep domain. If we imagine north to be in the $+y$ direction, each robot has five actions to choose from: moving east, northeast, north, northwest, and west. Initially, we allowed robots to choose any of these actions but we found that, when faced with the prospect of breaking the perimeter, they stopped moving and chose to remain deadlocked rather than make progress across the environment at high cost. So, in order to ensure progress across the area, a robot now chooses between moving northeast, north, and northwest when there is free space ahead. When

an obstacle blocks its path, it chooses between moving east and west. It chooses its actions based only on its expectations of its neighboring two teammates' actions because it is only with these two that perimeter breaks are possible. Since a robot cannot accurately estimate the actions of its teammates (it cannot know what constraints the teammate faces from its other neighbor), it selects the action that is most valuable assuming its teammates are equally likely to move northeast, north, or northwest.

6.1.2 P-MVERT

MVERT has the obvious disadvantage that robots act myopically. That is, they can only look ahead a single step so they cannot recognize when a perimeter break will occur until it is one step away. By then, it is too late to avoid it. We have improved MVERT to allow extended planning, resulting in what we have dubbed "P-MVERT" for Planning MVERT. Here, a robot first generates a set of candidate plans for itself from which it must choose one. It also generates a set of possible plans for each neighboring teammate that is representative of all the paths the teammate might take. It chooses for itself the plan that has the highest expected profit given a uniform distribution over the set of teammates' paths. We do not permit a more informed distribution to ensure that the performance difference between P-MVERT and M-MVERT is strictly a product of planning.

6.1.3 PC-MVERT

In P-MVERT, robots must guess their teammates plans. However, by allowing communication between robots, we can remove the guesswork. In PC-MVERT (Planning and Communicating MVERT), each time a teammate selects a plan, it broadcasts this plan to its neighbors. Its neighbors then use this information to update their own plans, allowing them to both plan around perimeter breaks and to react to their teammates intended actions. One may recognize that PC-MVERT is identical to the passive coordination mechanism in Hoplites. We have observed that PC-MVERT occasionally suffers from the instability we predicted in Section 4.2 for the passive coordination phase of Hoplites. When two adjacent robots' planning cycles become synchronized, they may simultaneously change their plans and break the perimeter. They may also oscillate between two distinct plans in an effort to respond to each other's last broadcast plans.

6.1.4 Hoplites

Finally, by adding the active coordination mechanism, we arrive at Hoplites itself. We compare each of these four approaches to demonstrate exactly what roles planning, communication, and active coordination play in the performance of Hoplites. It is worth mentioning that our initial implementation of Hoplites uses simplified versions of both the commitment and replanning protocols we described in Section 4.5.2 and Section 4.5.3. Firstly, two robots A and B working on a team plan P_{AB} are committed fully for the duration of P_{AB}. However, either robot may change its own actions in P_{AB} provided that the change is no more costly to its teammate than the original plan.

So, each can replan locally to accommodate new changes in its environment without having to confer with the other. Secondly, we do not allow a robot to consider offers from other teammates if they are currently committed. That is, if C makes an offer to B, B cannot accept that offer until its commitments to A are complete.

6.2 Simulations

We have tested our approach in a graphical simulation of five robots tasked with clearing an environment. We generated twenty environments of size 200×200 units, each with twenty obstacles. The obstacles are randomly placed and have randomly chosen dimensions ranging from 5×5 units to 15×15 units. An example environment can be seen in Figure 3 *(a)*. We ran each approach on each environment fifty times.

The robots have no prior knowledge of the number or configuration of the obstacles in the environment. They are equipped with simulated 360° laser scanners with a range of 200 units to provide sensory information as they move. Furthermore, each robot functions as an independent software agent. Robots communicate via UDP and are therefore subject to the speed and fidelity of this protocol. Lastly, in simulation, we prescribe that a robot moves at a rate of one unit per second; this roughly scales one map unit to one meter.

To make the domain challenging, we want robots to perform a sweep in which security is of the utmost importance. Specifically, we encourage robots to travel up to fifty units to avoid breaking the perimeter. We penalize a robot $\$500$ for each step that either breaks a previously-secure perimeter segment or fails to secure a previously-broken perimeter segment. In contrast, we only charge the robot $\$5$ per unit of time required to traverse the environment and $\$5$ per unit traveled. Each robot also receives $\$40$ ($1/5$th of the room's width) for each unit it moves towards the far end of the environment. So, if a robot moves from location (x_1, y_1) at time t to (x_n, y_n) at time t' via a path $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, its profit p is computed by:

$$
\begin{array}{ll}
p = 40(y_n - y_1) & \text{Revenue} \\
\quad -500 \sum_{i=1}^{n-1} \text{Broken}(x_i, y_i, x_{i+1}, y_{i+1}) & \text{Penalty for a Perimeter Break} \\
\quad -5 \sum_{i=1}^{n-1} \sqrt{(x_i - x_{i+1})^2 + (y_i - y_{i+1})^2} & \text{Cost of Travel} \\
\quad -5(t' - t) & \text{Cost of Time}
\end{array}
$$

where Broken$(x_i, y_i, x_{i+1}, y_{i+1})$ is a function that returns 1 if the perimeter is broken during the transition and 0 otherwise.

6.3 Results and Discussion

Table 1 summarizes the performances of MVERT, P-MVERT, PC-MVERT, and Hoplites. We measure performance in terms of the total number of perimeter breaks, the computation time spent by a single robot, and the distance traveled by a single robot. The first and second columns list each approach and the number of successful runs for each approach, respectively. The primary cause of run failure is that occasionally, a few robot processes will fail to attach to a network port, thereby causing failure in the

Algorithm	Runs	$\overline{B}$	$\text{SEM}_{\overline{B}}$	$\text{Rel}_{\overline{B}}$	$\overline{T}$	$\text{SEM}_{\overline{T}}$	$\text{Rel}_{\overline{T}}$	$\overline{D}$	$\text{SEM}_{\overline{D}}$	$\text{Rel}_{\overline{D}}$
MVERT	963	41.9	0.43	–	12.1	0.29	–	195.8	0.5	–
P-MVERT	973	22.4	0.51	0.54	29.2	1.72	2.41	200.4	0.53	1.02
PC-MVERT	957	8.6	0.20	0.39	20.1	0.36	0.69	198.8	0.54	0.99
Hoplites	966	5.5	0.15	0.63	28.9	0.90	1.44	208.6	1.44	1.05

Table 1: The relative performances of the four approaches. We ran each approach fifty times on each of twenty environments; failed runs are excluded from these results. We use three performance metrics: the number of perimeter breaks caused (B), the computation time spent by a single robot (T) measured in seconds, and the distance traveled by a single robot (D) measured in map units. For each measure, the first column is the mean, the second is the standard error of the mean, and the third is the mean of the current approach relative to the previously listed approach ($\frac{CURR}{PREV}$).

initialization sequence. We did not rerun trials because of time constraints (each run takes approximately five minutes). We have exclude these runs from the results.

$\overline{B}$ is the mean of the total number of perimeter breaks incurred by the team, $\text{SEM}_{\overline{B}}$ is the standard error of this mean, and $\text{Rel}_{\overline{B}}$ is the number of perimeter breaks incurred by the current approach relative to the previously listed approach. This measure reflects the team's ability to perform a secure sweep. As we would expect, each incremental increase in the capabilities of MVERT creates a significant corresponding decrease in the number of perimeter breaks.

$\overline{T}$ is the mean of the computation time in seconds spent by a single robot, $\text{SEM}_{\overline{T}}$ is the standard error of this mean, and $\text{REL}_{\overline{T}}$ is the time taken by the current approach relative to the previously listed approach. It includes the time spent planning individual and team paths, updating map information, communicating state information, and negotiating with teammates. The addition of a planning mechanism in P-MVERT increases the computation time but the subsequent addition of communication in PC-MVERT reduces this again. Active coordination between the robots in Hoplites results in another increase in computation time.

$\overline{D}$ is the mean of the distance in units traveled by a robot, $\text{SEM}_{\overline{D}}$ is the standard error of this mean, and $\text{Rel}_{\overline{D}}$ is the distance traveled by a robot in the current approach relative to the previously listed approach. Planning in P-MVERT increases the distance robots travel, but the communication in PC-MVERT does not affect it. Robots travel the farthest when actively coordinating in Hoplites.

The improvement in performance between MVERT and P-MVERT reinforces the idea that this domain requires planning. The sequence of good local solutions created by a one-step lookahead does not add up to a good global solution. This planning results in an increase in computation time. It also results in an increase in the distance traveled since robots will take longer paths around particular obstacles to avoid breaking the perimeter.

In PC-MVERT and P-MVERT, robots use the same algorithm to generate the set of candidate paths through the environment. Generally, only a few of these paths appear truly promising. We have observed that these paths tend to be of the same length and in the same direction, but pass on different sides of obstacles. In P-MVERT, robots

can only guess which of these paths will be most profitable. However, in PC-MVERT, robots know precisely what their teammates will be doing. Because this information results in a significant improvement in the performance of the team, we can infer that knowing teammates' actions enables robots to choose correctly from those paths. This corroborates our claim that planned tight coordination is essential in this domain. Also, an increase in communication results in a decrease in computation time because a robot no longer needs to generate and then evaluate an entire set of plans for its teammates; instead, it can rely on the plans they broadcast. Finally, there is effectively no change in the distance traveled by the robots since both approaches generate candidate paths in the same way.

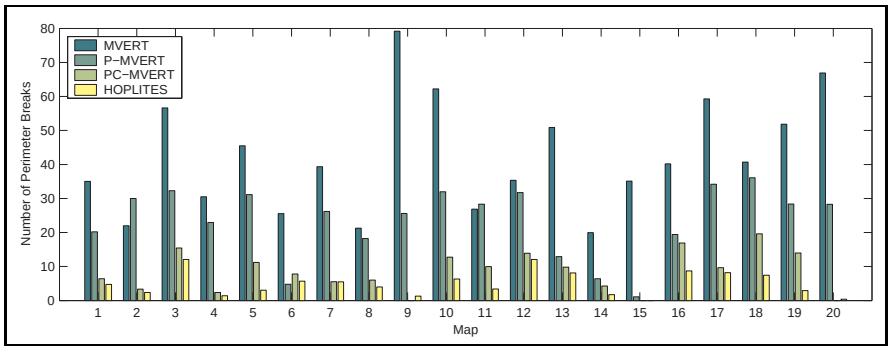
It is the performance improvement between PC-MVERT and Hoplites that most interests us. Firstly, it substantiates our claim that passive coordination alone may trap robots in local minima but that, by developing team plans and actively influencing each other's actions, the team can avoid many of these pitfalls. Secondly, it validates Hoplites. The results demonstrate that, through the market, Hoplites provides a powerful mechanism that allows robots to intentionally work together to find better solutions. There is an increase in computation time since developing team plans involves planning through high-dimensional joint action spaces. There is also an increase in distance traveled since the team plans that prevent perimeter breaks are usually long and more complex than the plans generated by P-MVERT and PC-MVERT.

The results in Table 1 combine the performance of the approaches over all the environments. In contrast, Figure 5 *(a)* presents the relative performances of each of the approaches on individual environments. It is clear from the data that both planning and communication improve the performance of MVERT; indeed, we expect nothing less. More interestingly, we observe that the effect of enabling explicit coordination differs between environments.

We would like to examine specifically how the effects of passive and active coordination change as we make environments more and more challenging. However, it can be difficult to determine by inspection alone which environments are more complex than others. We use the performance of PC-MVERT as a measure of complexity. Figure 5 *(b)* compares the performance of PC-MVERT and Hoplites. Environments are ordered by decreasing performance of PC-MVERT which correlates with increasing environmental complexity.

First, note that Hoplites outperforms PC-MVERT in nearly every environment. More importantly, the margin of improvement is significantly larger in more difficult environments, reaffirming that Hoplites is effective in facilitating the complex level of coordination required in complex tasks. Specifically, suppose we divide the environments into two sets of ten, split into an "easy" set and a "hard" set based on PC-MVERT's performance. Then, the number of perimeter breaks incurred on the easy set by Hoplites relative to PC-MVERT is 0.76 as compared to 0.54 on the harder set. This difference exists in some part because there is simply more room for improvement when PC-MVERT performs poorly than when it performs well. However, it is more importantly due to the design of Hoplites. Active coordination between robots can require extensive computation and create very complex interactions between robots. This complexity is most justified and most beneficial in correspondingly complex situations. Indeed, in the two instances in which PC-MVERT slightly outperforms Hoplites, the

(a)



(b)

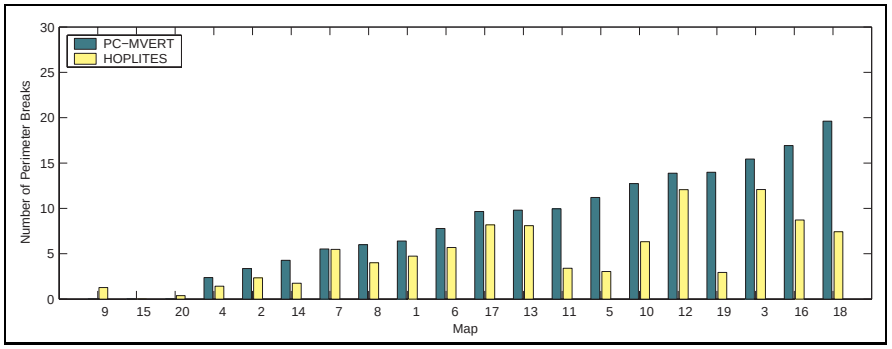


Figure 5: The performances on individual environments. *(a)* The number of perimeter breaks caused by MVERT, P-MVERT, PC-MVERT, and Hoplites on individual environments ordered by number. *(b)* The number of perimeter breaks caused by PC-MVERT and Hoplites on individual environments, ordered by increasing complexity. Error bars have been left out for clarity.

average number of perimeter breaks for both approaches is less than two, suggesting very easy environments.

7 Conclusion and Future Work

In this paper we have presented Hoplites, a sophisticated market framework for multi-robot teams performing tasks that require both planning and tight coordination. Hoplites consists of two different coordination mechanisms. In simpler situations, the team can complete its task faster by utilizing the passive coordination mechanism which facilitates more local decision making and is light on both computation and communication. More complex scenarios require more complex interaction between teammates. This is provided by the active coordination mechanism which enables robots to actively influence each other's actions over the market. Robots can dynamically choose the coordination mechanism that best fits the current needs of the task.

We have extensively compared Hoplites to the MVERT coordination framework

[5] in a security sweep domain. Hoplites dramatically outperforms MVERT in all instances of the domain. Hoplites maintains this advantage even after several significant improvements are made to MVERT. These results demonstrate that both planning and explicit tight coordination improve the quality of the team solution.

In the immediate future we plan to incorporate the negotiation and replanning protocols we described in Sections 4.5.2 and 4.5.3. We expect that this will provide increased flexibility in adapting commitments between teammates to changes in both the environment and the needs of the task. We also intend to demonstrate our work on a team of outdoor mobile robots. In the longer term, we hope to apply Hoplites to other domains such as constrained exploration and the art gallery problem. As well as providing additional validation of our framework, this will us provide the opportunity to strengthen different aspects of Hoplites.

Acknowledgments

This work was sponsored by the U.S. Army Research Laboratory, under contract "Robotics Collaborative Technology Alliance" (contract number DAAD19-01-2-0012). 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 of the U.S. Government.

References

[1] R. M. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Market-driven multi-robot exploration," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-02-02, January 2002.

[2] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2000.

[3] R. M. Zlot and A. Stentz, "Market-based multirobot coordination using task abstraction," in *Proceedings of the International Conference on Advanced Robotics (FSR)*, July 2003.

[4] E. Østergaard, M. J. Matarić, and G. S. Sukhatme, "Distributed multi-robot task allocation for emergency handling," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2001.

[5] A. Stroupe, "Collaborative execution of exploration and tracking using move value estimation for robot teams (MVERT)," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 2003.

[6] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multi-robot coordination," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 758–768, Oct. 2002.

[7] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-uav context," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2004.

[8] S. S. C. Botelho and R. Alami, "A multi-robot cooperative task achievement system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2000.

[9] L. Chaimowicz, T. Sugar, V. Kumar, and M. Campos, "An architecture for tightly coupled multi-robot cooperation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2001.

[10] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, A. Casal, and A. Baader, "Force strategies for cooperative tasks in multiple mobile manipulation systems," in *Robotics Research: The Seventh International Symposium*. Springer, 1996, pp. 333–342.

[11] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith, "First results in the coordination of heterogeneous robots for large-scale assembly," in *Proceedings of the International Symposium on Experimental Robotics (ISER)*, December 2000.

[12] D. J. Naffin and G. S. Sukhatme, "Negotiated formations," in *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, March 2004.

[13] H. G. Nguyen, N. Pezeshkian, M. Raymond, A. Gupta, and J. M. Spector, "Autonomous communication relays for tactical robots," in *Proceedings of the International Conference on Advanced Robotics (ICAR)*, June 2003.

[14] A. Wagner and R. Arkin, "Multi-robot communication-sensitive reconnaissance," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2004.

[15] P. Caloud, W. Choi, J. C. Latombe, C. L. Pape, and M. Yim, "Indoor automation with many mobile robots," in *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, July 1990.

[16] B. L. Brumitt and A. Stentz, "Dynamic mission planning for multiple mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 1996.

[17] R. G. Brown and J. Jennings, "A pusher/steerer model for strongly cooperative mobile robot manipulation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, August 1995.

[18] M. J. Matarić, "Behavior based robotics," in *MIT Encyclopedia of Cognitive Sciences*. MIT Press, 1999.

[19] E. Bahceci, O. Soysal, and E. Sahin, "A review: Pattern formation and adaptation in multi-robot systems," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-03-43, October 2003.

[20] J. Fredslund and M. J. Matarić, "Robot formations using only local sensing and control," in *Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, July 2001.

[21] H. Yamaguchi, "Adaptive formation control for distributed autonomous mobile robot groups," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 1997.

[22] L. E. Parker, "Designing control laws for cooperative agent teams," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 1993.

[23] T. Balch and M. Hybinette, "Social potentials for scalable multirobot formations," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2000.

[24] S. S. C. Botelho and R. Alami, "M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 1999.

[25] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, August 2000.

[26] R. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Transactions on Computers*, vol. C-29, no. 12, December 1980.

[27] A. Stentz and M. B. Dias, "A free market architecture for coordinating multiple robots," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-99-42, December 1999.

[28] M. B. Dias, R. Zlot, M. Zinck, J. P. Gonzalez, and A. Stentz, "A versatile implementation of the traderbots approach for multirobot coordination," in *Proceedings of the International Conference on Intelligent Autonomous Systems (IAS)*, March 2004.

[29] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[30] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Department, Iowa State University, Tech. Rep. 98-11, October 1998.

[31] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1999.

[32] K. Mehlhorn, *Data structures and algorithms 3: multi-dimensional searching and computational geometry*. Springer-Verlag, 1984.

[33] M. van Kreveld, "Variations on sweep algorithms: efficient computation of extended viewsheds and class intervals," in *Symposium on Spatial Data Handling*, August 1996.

[34] T. Parsons, "Pursuit-evasion in a graph," in *Theory and Applications of Graphs*. Springer-Verlag, 1976.

[35] I. Suzuki and M. Yamashita, "Searching for a mobile intruder in a polygonal region," *SIAM Journal on Computing*, vol. 21, no. 5, pp. 863–888, 1992.

[36] A. Efrat, L. J. Guibas, S. Har-Peled, D. C. Lin, J. S. B. Mitchell, and T. M. Murali, "Sweeping simple polygons with a chain of guards," in *Proceedings of the Symposium on Discrete Algorithms*, January 2000.

[37] L. H. Tseng, P. J. Heffernan, and D. T. Lee, "Two-guard walkability of simple polygons," *International Journal of Computational Geometry and Applications*, vol. 8, no. 1, pp. 85–116, 1998.

[38] C. Icking and R. Klein, "The two guards problem," in *Proceedings of the ACM Symposium on Computational Geometry*, June 1991.

[39] L. J. Guibas, J.-C. Latombe, S. M. LaValle, D. Lin, and R. Motwani, "Visibility-based pursuit-evasion in a polygonal environment," in *WADS '97 Algorithms and Data Structures (Lecture Notes in Computer Science, 1272)*. Springer-Verlag, 1977.

[40] L. Guilamo, B. Tovar, and S. M. LaValle, "Pursuit-evasion in an unknown environment using gap navigation graphs," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2004.

[41] B. P. Gerkey, S. Thrun, and G. Gordon, "Visibility-based pursuit-evasion with limited field of view," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2004.

[42] S. M. LaValle, B. H. Simov, and G. Slutzki, "An algorithm for searching a polygonal region with a flashlight," *International Journal of Computational Geometry and Applications*, vol. 12, no. 1-2, pp. 87–113, 2002.

[43] J.-H. Lee, S. Y. Shin, and K.-Y. Chwa, "Visibility-based pursuit-evasion in a polygonal room with a door," in *Proceedings of the ACM Symposium on Computational Geometry*, June 1999.

[44] J. Kim, R. Vidal, D. Shim, O. Shakernia, and S. Sastry, "A hierarchical approach to probabilistic pursuit-evasion games with unmanned ground and aerial vehicles," in *Proceedings of the IEEE Conference on Decision and Control*, December 2001.