A Market-Based Framework for Tightly-Coupled Planned Coordination in Multirobot Teams

Nidhi Kalra

CMU-RI-TR-06-57

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

January 11, 2007

Thesis Committee:
Anthony (Tony) Stentz, Chair
M. Bernardine Dias
Manuela Veloso
Gaurav Sukhatme, University of Southern California

Contents

List of Figures

List of Tables

Abstract

This dissertation explores the challenges of one of the most difficult classes of real-world tasks for multirobot teams: those that require long-term planning of tightly-coordinated actions between teammates. These tasks involve solving a distributed multi-agent planning problem in which the actions of robots are tightly coupled. Moreover, because of uncertainty in the environment and the team, robots must frequently replan and closely coordinate with each other throughout execution.

We have developed a coordination framework called Hoplites in response to the need for effective approaches to these problems. Although planning for tightly-coupled multirobot systems is a difficult problem, Hoplites solves this problem efficiently by using distributed decision-making whenever possible and centralized planning as required. Hoplites is a market-based system that consists of passive coordination and active coordination mechanisms which are tailored to easier and harder problem scenarios, respectively. Passive coordination is light on computation and communication and allows teammates to iteratively respond to each other's actions without directly influencing them. When passive coordination traps robots in local minima, active coordination improves solutions by enabling robots to influence each other directly by buying each other's participation in complex plans over the market. Because Hoplites selectively injects pockets of complex coordination into the system, it provides these improvements while remaining computationally competitive with other distributed approaches. Moreover, this selective complexity allows Hoplites to outperform centralized approaches as well because it can often exploit planners with performance guarantees which a centralized approach cannot. Additionally, Hoplites is widely applicable to real-world problems because it is general, computationally feasible, scalable, operates under uncertainty, and improves solutions with new information.

This dissertation makes a number of contributions to the literature. First, it develops Hoplites, a general and adaptive approach to these complex problems. Second, it formalizes the problem space which, in turn, enables us to describe and share solutions between domains that may have previously appeared disparate. Third, it is the first direct application of market-based approaches to tight coordination. Fourth, it presents the first evaluation of and recommendations for planning algorithms for tight coordination. Lastly, it improves the previous state-of-the-art coordination framework for these problems.

Funding Sources

This research was sponsored by the U.S. Army Research Laboratory 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.

Acknowledgements

First, I would like to thank my adviser Tony Stentz for his invaluable guidance and support in all of my endeavors, for being an inspirational researcher, and for making my Ph.D. a rich and rewarding experience that I will look back on with great fondness. I would also like to thank my committee members Manuela Veloso, Bernardine Dias, and Gaurav Sukhatme for their insight and assistance, particularly in the final months of this thesis.

My gratitude also extends to my colleagues for their collaborative efforts and advice: Dave Ferguson for his invaluable contributions throughout my thesis, Alcherio Martinoli for showing me a fascinating new side of multirobot coordination, members of the R-CommerceLab (Tony Stentz, Bernardine Dias, Rob Zlot, Gil Jones, and Dave Ferguson) for their insight into all things market-based, and my qualifying committee (Manuela Veloso, Martial Hebert, and Jeremy Kubica) for their recommendations in the early days of this work. Special thanks to Marc Zinc and Freddie Dias for their hard work in keeping our robots running strong, and, with Dave Ferguson, Gil Jones, Ayorkor Mills-Tettey, Juan Pablo Gonzalez, Eugene Marinelli, Max Likhachev, Boris Sofman, and Ling Xu for helping with the robot experiments even in the most adverse conditions.

Thanks also to Suzanne Lyons Muth and Sumitra Gopal for all their hard work behind the scenes and to my office mates Juan Pablo Gonzalez and Brenna Argall for great conversations and lots of laughs.

My time at the Robotics Institute has been enriched by so many wonderful friends; I cannot name you all. But thanks for all the office antics, nights on the town, dinners at My Thai, barbecues, failed camping trips, protest marches, movie nights, and so much more. Its been more fun than I ever imagined.

I would also like to thank my grandparents, my parents Rajiv and Seema, and my sister Anita for their love and support. None of this would have been possible without you.

Lastly, thanks most of all to Dave and Oscar, my team.

For my family

Chapter 1

Introduction

Like humans, robots accomplish much more by working in teams than by working alone. They can often achieve the same tasks more efficiently and robustly. More importantly, they can accomplish a wide range of valuable tasks that are simply impossible for a single robot.

For example, consider using autonomous rovers to explore a hazardous planetary environment. For robustness, information dissemination and retrieval, security, or mission re-tasking, we require each rover to always remain within communication contact with some base station. This task is an instance of constrained exploration, and it would be impossible for a single rover working alone because its exploration radius would be severely limited. Instead, we would employ a coordinated team where each member could communicate with the base station either directly or by using its teammates as relay nodes. In essence, the rovers must closely coordinate to maintain an ad-hoc network as they navigate through the environment. For missions such as this, successful coordination is critical for successful mission completion, yet it is especially difficult to achieve.

This thesis explores the coordination challenges posed by problems such as this that require planned tight coordination between teammates throughout execution. These tasks involve solving a distributed multi-agent planning problem in which the actions of robots are tightly coupled. Because of uncertainty in the environment and the team, these problems also require persistent tight coordination between teammates throughout execution.

1.1 Applications

There are a number of applications for which a single robot will not suffice and several are required. These can be categorized into those tasks that require simultaneous execution of independent components, multiple functionalities, and coordinated execution. Some tasks

may have combinations of these features[1].

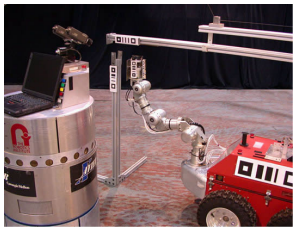
Tasks requiring simultaneous execution of independent components.

In reconnaissance, security, and urban search and rescue missions, we may want to observe the same area of interest from different vantage points at the same time. In demolition tasks, detonations may need to occur simultaneously to ensure a safe collapse. In these and similar scenarios, multiple robots are required to simultaneously perform several independent activities from different locations.

Unmanned ground vehicles on patrol [1].

Multi-functional tasks.

Large-scale construction tasks typically involve several steps. For example, materials and equipment must be transported from a drop-off site to the construction site, they must be placed carefully in position, and the parts must be assembled. Hazardous clean-up tasks also frequently involve several activities such as monitoring the ambient temperature and the toxicity of the atmosphere, detecting and cleaning spills, and quarantining hazardous areas. Completing each of these tasks requires several different functionalities. A heterogeneous team of simple yet specialized robots may be more effective than a single overly-complex robot that can do no individual task well. Indeed, a team is necessary when no single individual robot has all the required capabilities.



Robots dock a beam [2] and move HAZMAT waste [3].

Tasks requiring tightly-coordianted execution.

In many tasks, the subcomponents of the task cannot be completed independently by individual robots. Instead, multiple teammates must coordinate very closely throughout execution to achieve the team mission. By working together, teammates can increase the team's physical capabilities, safety, communications, and visibility.

[1] Robot images in this section have been provided by other researchers: ground vehicle photo courtesy of Hoa Nguyen, beam docking photo courtesy of Reid Simmons, HAZMAT robot photo courtesy of James Melton, beam transportation and cliff robot photos courtesy of Terry Huntsberger, formation photo courtesy of Ron Arkin, robot connectivity photo courtesy of Hoa Nguyen, and pursuit robots photo courtesy of Shankar Sastry.

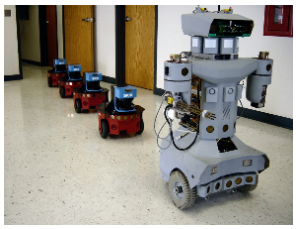
Increased physical ability. Many domains require the transport of objects such as tools, beams, barrels, and crates from one location to another. These domains include the construction and clean-up tasks we described earlier, as well as warehouse supply, shipping and delivery, and demolition. Often, objects are too large to be manipulated by a single robot, but several coordinating robots can do the job successfully.

Two robots transport a beam [4].

Increased safety. For human safety, we would like to use robots in hazardous environments such as mines, caves, and cliffs for missions such as exploration, mapping, and reconnaissance. In these environments, the liklihood of robot loss can be very high because, in addition to the dangers posed by the environment, human operators may not be able to retrieve or repair a robot in case of malfunction. A robot team may be able to use coordinated tactics to accomplish the tasks more safely. Consider a single robot climbing the side of a cliff; as with human climbers, a slip or a misstep can lead to disaster. Team strategies such as belay systems in which robots assist each other can be much more robust to accidents. Similarly, bounded over-watch tactics or movement in formation can be safer strategies in hostile areas than independent navigation.

Robots scale a cliff [5] and move in formation [6].

Improved communication. In hazardous environments, robots also frequently lose communication to external points of contact such as human operators and control stations. Without communication, it can be difficult to detect dangerous situations, diagnose and recover from failures, and observe mission progress. With a team, robots can act as relays between each other and the team's operators and thus decrease the time without communication access.



Robots provide network connectivity [7].

Increased visibility. Many surveillance, security, and reconnaissance tasks require maximizing visibility and securing areas. Security sweep, for instance, involves making a pass through an area that maximizes visibility and maximizes the likelihood of detecting any adversaries. General monitoring tasks may require that portions of an area be constantly monitored while other areas are periodically monitored. In both of these cases, complex environments significantly reduce visibility and require the coordination of multiple robots throughout execution.

A team of ground and aerial robots pursue an evader [8].

1.2 Thesis problem space

In this thesis, we are interested in solving complex instances of the class of tasks that require tightly coupled execution. As illustrated in Figure 1.1, we can describe the problem space more formally by arranging multirobot tasks along three axes. The first two measure the degree of coordination between teammates and the degree of planning required to solve the problem effectively. The third axis measures whether or not the problem can be captured as a set of coupled local problems.

Although the degree of coordination and the degree of planning are continuous, we can describe our problem space precisely with two rules. Consider a problem T that may require robots to interact during execution. First, on the coordination axis, if the execution-time coordination required between robots can be successfully resolved during task allocation, then T is *not* in our problem space. Second, on the planning axis, if the execution-time coordination between robots can be planned using hill-climbing, then again T is *not* in our problem space. Problems in our space satisfy the inverse of both of these rules: they require execution-time coordination that cannot be resolved during task allocation *and* that cannot be planned using hill-climbing. If, despite the challenges posed by these features, these problems can simultaneously be modeled with a set of locally manageable constraints (as indicated on the third axis), then they can potentially be solved using distributed approaches such as the one this thesis develops. Lastly, for brevity, we approximate the coordination and planning requirements through the remainder of this thesis with the terms "tight coordination" and "long-term planning", respectively; as we describe in this section, these are useful but imprecise labels.

Loose vs. tight coordination

Different multirobot tasks can require significantly different degrees of interaction between robots in order to be completed effectively. At one end of the spectrum are tasks that

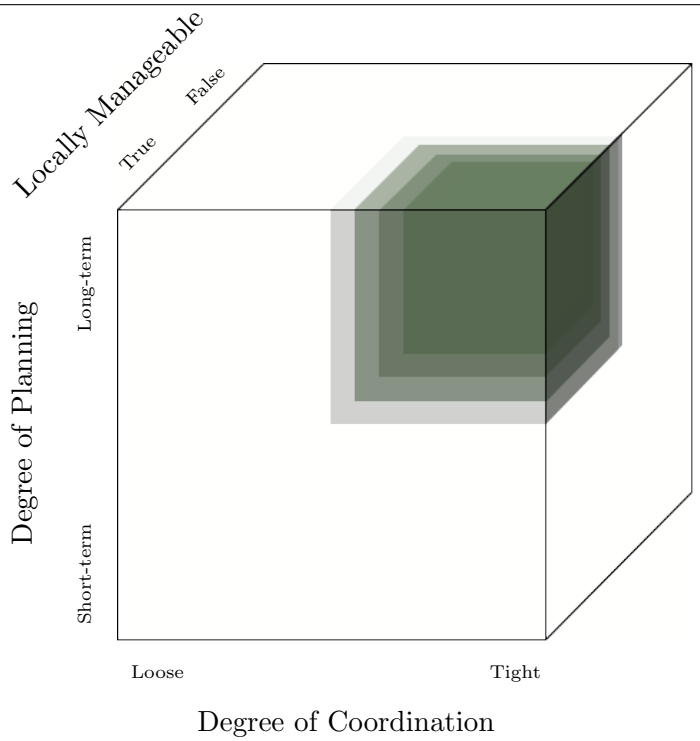


Figure 1.1: The problem space addressed by this thesis is the class of multirobot tasks that require tight coordination and long-term planning but that can be described by a set of locally manageable constraints.

could be completed by an individual robot but which a team can accomplish faster and more robustly by breaking the task into a set of independent components and distributing them to different teammates who complete the components in parallel. Such tasks include exploration and mapping [9, 10], reconnaissance [11], emergency handling [12], hazardous cleanup [13], and tracking [14, 8]. For instance, a single robot could map a multi-storied building, but, by decomposing the task into individual floors and assigning one robot per floor, a team can complete it much faster. This distributes both the planning load and the execution load across the team. Moreover, the team is robust through redundancy: if one robot fails, another can complete its mission components.

Such tasks require coordination within the team to decompose the task into subtasks and allocate the subtasks to individuals. The capabilities and states of several robots may need to be considered simultaneously in order to make the best decomposition and allocation decisions. However, once a robot has been assigned a subtask, it interacts minimally with its teammates because its subtask can be completed independently. A team of robots with this level of interaction between teammates is termed *loosely coordinated*. From a coordination perspective, research is aimed at developing decomposition and allocation techniques that are fast and most effectively use the team's resources.

At the other end of the spectrum, the tasks we are interested in solving require a very high degree of interaction between team members throughout execution because the actions available to one team member depend significantly on the simultaneous actions of its teammates. Such tasks include the cooperative manipulation of objects [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 5], playing team games [25, 26], moving in formation [27, 28], surveillance and monitoring [29, 30], and moving with communication constraints [7, 6, 31]. Firstly, unlike the tasks we discuss earlier, these cannot easily be decomposed into independent subtasks. To illustrate, suppose we task a team of robots with jointly carrying a heavy object. Then, we cannot split the object into pieces and distribute them to the team for independent transport. Instead, the robots must execute the task together. Secondly, these tasks usually impose constraints between teammates. In the case of transporting heavy objects, the teammates must remain within a certain distance of each other and move in roughly the same direction to avoid dropping the object. Thirdly, the actions of one robot can have significant consequences for its teammates and the task. For instance, if even one robot moves too quickly or in the wrong direction, the team can be thrown off balance and drop the object. Uncertainty in the environment, sensors, and actuation, makes it necessary for the robots to communicate constantly to ensure a successful joint trajectory. In total, these tasks require a much higher degree of interaction between teammates than those we discussed earlier. Robots must frequently reevaluate their actions within the context of their teammates' simultaneous actions. A team with such close coupling between its members is called *tightly coordinated*.

Researchers developing tightly coordinated teams face significantly different challenges from those working on moderately or loosely coordinated teams. Firstly, they must develop coordination techniques that facilitate constant, fast interaction between teammates and their environment. Secondly, they must accommodate the potentially heavy demand for information between teammates. Furthermore, because of the tight coupling of actions, tightly coordinated teams cannot easily take advantage of the distributed planning and execution methods that make loose coordination tractable. Thus, thirdly and most importantly, researchers must develop coordination and planning techniques that successfully solve the joint planning problem while still being tractable. We explore this planning aspect in the next section.

The tasks we describe are extreme points on the spectrum of coordination demands: one set requires no interaction between teammates during execution while the other set requires constant interaction. Nevertheless, many tasks call for something in between. For example, a mission involving precedence-constrained subtasks may require robots to periodically interact during execution to determine when they can begin the different subtasks [32, 33]; such a team might be termed "moderately" coordinated. Given this continuous range, we have developed a rule to more strongly define our problem space: if, for a particular task

T, the execution-time coordination required between robots can be successfully resolved during task allocation, then T is *not* in our problem space.

For example, MacKenzie [34] uses this technique to address the problem of robots completing a set of subtasks that are constrained to begin within a short time window. The challenge of determining the start window during execution is circumvented by incorporating it into the task allocation phase: subtasks are allocated based on when a robot could actually complete the task and thus the start time is established before execution. This task is not in our problem space. In contrast, the coordination requirements of the constrained exploration problem that we introduced at the very beginning of this chapter cannot be solved during task allocation. (In this problem, the team of robots must explore an area while maintaining contact with a base station.) It is conceivable that in limited cases of this problem, one could preplan a set of routes that meet these constraints and then allocate them as tasks before execution [35]. In realistic scenarios however, the environment is unknown (hence the need for exploration), and thus the preplanned paths are unlikely to maintain those constraints or even be traversable. For domains such as this, approaches that attempt to solve execution-time coordination during task allocation will be brittle and ineffective. Indeed, for domains such as collaborative manipulation, this technique is impossible because the interactions between robots are so tightly coupled and sensitive to uncertainty that they cannot be captured in advance.

Short-term vs. long-term planning

The problems that remain in our space require robots to coordinate significantly during execution. Our second axis evaluates how far in advance robots determine their interactions before executing those interactions. This lookahead is called the *planning horizon*. In the multirobot literature, tightly coordinated robot teams have almost exclusively been employed to complete tasks that involve simple interactions between teammates. That is, the interactions between robots and the responses they have to each other's activities are typically easily described, do not change, and require little advanced planning because the constraints imposed by the tasks are fairly simple. For example, when moving in formation, a robot knows a priori the position it ought to maintain relative to its teammates. So, when a teammate moves, the robot's own heading and velocity can be computed straightforwardly. Thus, coordination can often be achieved with methods that use short-term planning such as reactive and behavior based approaches which we discuss further in Chapter 2. These techniques have been applied to a number of domains including the cooperative manipulation of objects [15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 5], movement in formation [27, 28], and movement with communication constraints [7, 6]. By successfully using short planning horizons, tight coordination is tractable for most domains investigated by researchers.

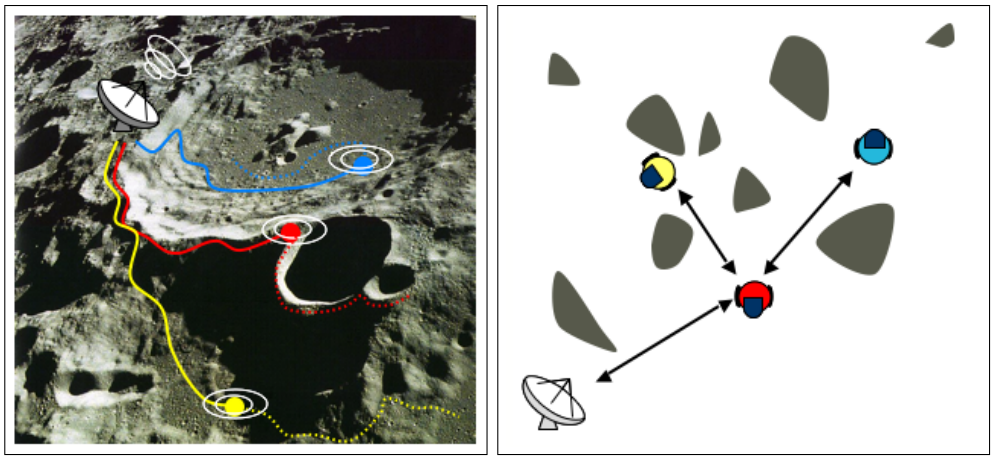


Figure 1.2: Constrained exploration is one task that falls in our problem space. In this motivating example, a team of three lunar rovers *(left)* must maintain contact with a base station to Earth. This problem can be illustrated by a simplified diagram *(right)* where lines between robots indicate communication connectivity and the obstacles preventing communication are shaded in grey.

The tasks we would like to perform, however, have significantly more complex constraints between robots. Consider the exploration problem we described in which robots must gather information about the environment while maintaining communication with a base station by using each other as relay nodes. To explore, a robot must follow a path that satisfies these constraints, perhaps with the assistance of its teammates. In cluttered environments, the teammates may need to traverse complex paths simultaneously to provide communication service to different members of the group. Constrained exploration is illustrated in Figure 1.2.

The gallery monitoring and security sweep domains impose similar constraints. In the former, robots monitor a complex environment and ensure that certain areas are continuously observed while others are periodically checked for intruders. The task may also require that robots handle on-line monitoring requests, for instance, if suspicious activity has been detected in an area. The robots must execute coordinated monitoring sequences that satisfy the constraints of the task. In the security sweep domain, robots sweep an area suspected to contain mobile adversaries. The robots must determine and execute in concert those paths through the environment that maximize the likelihood of detecting all adversaries. Security sweep is illustrated in Figure 1.3

In such missions, the interactions between teammates are complex and not easily described. Solving these problems means solving a multi-agent planning problem in which the actions of robots are tightly coupled. Additionally, the actions taken by the team can have significant impact on their ability to complete the task at a later time. Thus, the tasks we

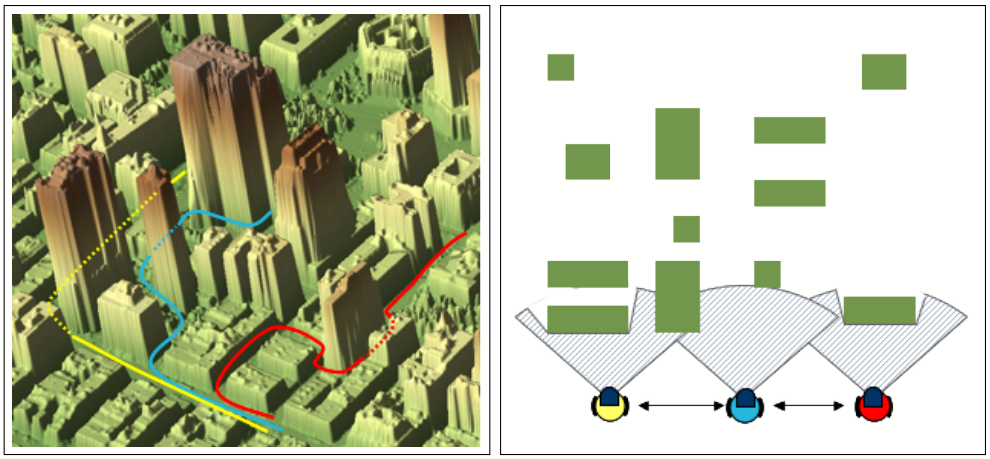


Figure 1.3: Security sweep is a second task that falls in our problem space. In this motivating example, a team of autonomous vehicles *(left)* must sweep an area of New York City. This problem can be illustrated by a simplified diagram *(right)* where lines between robots indicate the front of the sweep, the hatched areas indicate the robots' sensor ranges, and the green rectangles are obstacles around which they must sweep.

are interested in completing require long-term planning and are too complex to be solved with short-term techniques.

As with coordination requirements, planning requirements span a spectrum of possibilities, from seconds and minutes to hours and days. Moreover, "short-term" and "long-term" are imprecise labels: planning several minutes in advance many be considered long-term planning for nanorobots but short-term planning for large ground vehicles. We have developed a second rule to precisely define our problem space: if, for a particular task T, the execution-time coordination required between robots can be successfully planned using hill-climbing [36], then T is *not* in our problem space. Essentially, if a one-step coordination lookahead will not trap the team in a local minima, then these problems can already be solved using existing techniques.

Note that this rule does not refer to all types of planning required by a problem, only the planning of interactions between teammates. For example, suppose a task requires two follower robots to trail a leader robot as it moves to some target region in a cluttered environment. Although the leader robot may use an optimal path planner to find a route, this problem does not fall into our problem space because the follower robots can successfully use hill climbing to follow the leader. In contrast, in constrained exploration, hill climbing can easily trap robots in local minima: the team can reach a configuration in which further exploration is impossible without breaking communication constraints; this configuration could be avoided by planning the interactions between teammates in advance.

Locally manageable constraints

We can further differentiate the problems that remain in our space by whether or not the global constraints imposed by the problem can be locally managed. By "global constraints" we are referring to the constraints that must be met by the team as a whole. We say that these constraints can be locally managed if we can produce a (perhaps unique) set of constraints that

1. for each robot involve a strict subset of its teammates and that
2. satisfy the global constraint when they are all met.

A problem with coordination requirements that can neither be resolved during task allocation nor be planned using hill climbing is already very challenging to solve. If, in addition to these features, its global constraints cannot be decomposed into local constraints, then it effectively requires a centralized approach because no decoupling of the problem is possible.

Consider an instance of constrained exploration that requires every robot to maintain direct line-of-sight connectivity with every teammate throughout the duration of the mission. We cannot create a set of local constraints because each robot must continuously consider the actions of every teammate in order to evaluate its connectivity and choose future actions. The large information requirements of such problems make instances involving large teams infeasible. Simultaneously many benefits typically associated with distributed approaches (e.g. localized information, scalability, and responsiveness), cannot be harnessed well. In sum, it may often be preferable to use a centralized approach which at least minimizes the communication load and redundancy in information. The tradeoff is that these approaches tend to suffer from single points of failure and cannot distribute computation.

Moreover, for these problems, even evaluating the team's solution becomes impractical (if not impossible) in real implementations. To evaluate the team's connectivity on Mars, an observer on Earth must have accurate information about each robot's location and a map of the environment and, for every time step, must solve a connectivity problem that is $\mathcal{O}(n^2)$ in the number of robots. This must additionally occur fast enough to keep up with the robots' movement through the environment. Such problems do not scale to large teams and can become intractable.

Alternatively, consider the instance of constrained exploration in which every teammate must maintain line-of-sight contact with some base station. In the most general case, this problem also requires each robot to continuously consider every one of its teammates actions. Suppose that a robot used some routing of teammates to communicate with the base station. Then if there was a break in that particular route, the robot would have to evaluate the positions of every team member to determine if it was connected. Moreover, to plan future routes, the robot would require information about every other teammates'

future positions as well. As with the previous problem, this general formulation cannot be solved in a distributed way and solutions cannot be evaluated in practice; thus, this becomes an infeasible problem.

Fortunately, this problem is locally manageable. For example, we can order each robot on the team from r_0 to r_{n-1} and create a local constraints c_i for each robot r_i:

- $c_0 : r_0$ must be connected to the base station
- $c_j : r_j$ must be connected to its teammate r_{j-1} for $0 < j < n$

Then if all constraints are met, the team must be a connected network. Now, each robot r_i only needs to maintain contact with at most one teammate (r_{i-1}) which allows us to decompose the problem and make it tractable; but, this problem still requires tight coordination (between r_i and r_{i-1}) and planning of future actions so it remains in our problem space. Moreover, the problem can be evaluated efficiently since solution evaluation is linear in the number of robots.

Essentially, we are creating a set of constraints that are *sufficient but not necessary* for maintaining the global constraint (which in many cases is not even possible to evaluate). We are concerned with this reduced version of the problem that approximates the global problem and that may be solved with more distributed techniques.

1.3 Problem statement

To date, the multirobot research community has not adequately addressed the coordination needs of robot teams operating in complex problem domains such as security sweep, the gallery monitoring problem, and communication-constrained exploration. This thesis is aimed at identifying the coordination needs of such tasks and developing a general, flexible coordination framework that meets these needs. Such a framework should be applicable to many domains and should not restrict the number of teammates that can coordinate or the ways in which they can coordinate. It should be computationally feasible for real domain instances where time constraints may be very relevant. Furthermore, it should be functional in the presence of uncertainty and responsive to new and better information by repairing team solutions. In addition to being evaluated in simulation, this framework must be demonstrated and verified on real robot teams.

1.4 Thesis statement

This thesis asserts that using market-based techniques to selectively negotiate tightly-coupled plans enables multirobot teams to tractably produce high-quality solutions to challenging real-world problems that require planned, tight coordination between teammates.

1.5 Overview of approach

Core Approach

Although planning for tightly-coupled multirobot systems is a difficult problem, this difficulty can vary greatly between different instances of the same domain: it depends significantly on the complexity of the environment and the size of the team. For example, let us return to the constrained exploration problem of robots exploring a hazardous planetary environment while maintaining line-of-sight communication connectivity. In this domain, a cluttered environment is more challenging than an obstacle-free environment that permits unobstructed visibility and thus full network connectivity. Additionally, the same instance of a problem can be harder for a team of only five robots than for a team of fifty robots because the latter has greater visibility and thus greater connectivity. The difficulty of the task can also vary within a single instance of the domain. In environments with only one cluster of obstacles in a corner, the empty portions can be easily explored, but the cluttered corner may require close coordination between many teammates to ensure connectivity.

For efficiency, we want a coordination method that is only as complex as necessary to solve the problem at hand. That is, when robots are faced with an easy problem scenario, they should be able to coordinate with minimum communication and computation expense while still producing good solutions. Alternatively, in difficult scenarios, robots should be equipped with a complex coordination method that may require more resource expenditure to produce a good solution. The challenge, however, is the complexity of the problem cannot be known in advance or determined by inspection, and it may change during the course of a mission.

Our approach to this problem is a coordination framework in which the complexity and strength of the coordination adapt to the difficulty of the problem. We call our coordination framework "Hoplites"; the name is a reference to the ancient Greek infantrymen who specialized in complex, tightly-coordinated maneuvers. Robots begin with *passive coordination* which is light on communication and computation and which allows teammates to iteratively respond to each other's actions without directly influencing them. When passive coordination traps robots in local minima, *active coordination* improves solutions by enabling robots to influence each other directly and participate in complex, tightly-coupled plans. Active coordination consumes more resources but is also capable of producing much better solutions.

These features directly lead to the major contribution of this thesis: that Hoplites outperforms competing distributed approaches by enabling pockets of complex coordination between several robots that result in better global solutions. Moreover, by selectively injecting this complexity, Hoplites provides these improvements while remaining computationally

competitive with other distributed approaches. Finally, this selectivity also enables Hoplites to produce better solutions than centralized approaches which, for tractability, must resort to algorithms that produce feasible but very sub-optimal solutions. Hoplites, on the other hand, selectively uses these approaches and thus pays the price of significant sub-optimality only when necessary.

Approach Details

We have formulated Hoplites as a market-based approach to harness many of the associated benefits such as computational tractability, operation under uncertainty, flexibility, and generality. In typical market-based approaches, robots act as self-interested agents participating in a market economy; a robot receives revenue when it takes actions that further the team's mission (e.g. by exploring previously-unknown areas) and incurs costs when it consumes team resources (e.g. fuel). Each robot attempts to maximize its individual profit by trading tasks over the market using auctions. Because the robots' individual motivations are aligned with the team's goals, this process of redistribution simultaneously results in efficient solutions to the problem.

Hoplites extends market-based approaches in two significant ways. Firstly, to facilitate tight coupling between teammates, the profit a robot receives through its actions depends on the simultaneous actions of its teammates via a penalty mechanism. Secondly and more importantly, instead of buying and selling tasks as in other market-based approaches, robots buy and sell action-level plans that enable complex coordination.

Hoplites uses two different coordination mechanisms to respond dynamically to easier and harder problem scenarios; each uses market-elements to facilitate coordination. The first, passive coordination, is designed for simpler scenarios and enables the team to work faster by facilitating more local decision making and coordination that is relatively light on both communication and computation. Throughout execution, robots periodically broadcast their future actions. A robot passively coordinates with its teammates by replanning its own actions to produce a more profitable plan once it knows its teammates' projected actions. It then re-broadcasts any changes to its plan, which allows its teammates to respond in turn. For example, a robot r performing constrained exploration might first plan a direct path to an unexplored area. Upon learning that its teammate t's intended actions will leave it without communication access along a portion of this direct path and thus result in a costly penalty, it can alter its route to the area such that it will remain in contact and garner a larger profit. Thus, passively coordinating robots influence and coordinate their actions implicitly through changes in plans.

Harder scenarios may require more complex interactions between teammates which, in turn, require a more complex coordination mechanism. When actively coordinating, robots

try to influence each other's actions explicitly by buying their teammates' participation in complex plans over the market. For example, if the area that r wishes to explore is surrounded by obstacles, perhaps no action on its part alone will allow it to maintain line-of-sight contact with its teammate t. Then, it may need to explicitly alter t's path to ensure connectivity. So, r develops a candidate solution that involves both it and t, it proposes this plan to t, and then requests a price quote that reflects how much compensation t will require from r before it will participate in this plan. t considers the plan alongside its other options and responds with a price that can include costs for extra travel time, penalties for losing communication contact with other teammates, and even the opportunity cost of future actions. If t's cost is less than the financial benefit the plan provides to r (e.g. because exploring this area will generate a great deal of revenue), r will pay t and t will be bound to this contract. If not, then r may need to produce an alternative plan perhaps with other or additional teammates, abandon its efforts to explore the area, or continue to explore it and incur the penalty for losing contact. By encoding the benefit or cost of actions into the price, the market acts as a clearinghouse for plans and allows the team to efficiently trade-off demand from different parts of the group. With both coordination methods, Hoplites is able to very efficiently produce high-quality solutions to these complex problems.

Limitations

Hoplites is a powerful coordination framework and, as we show in this thesis, can successfully solve the domains in our problem space. Nevertheless, the improvements come at the expense of greater implementation complexity compared to competing approaches: robots must be able to consider incoming requests for active coordination, track current and future commitments, monitor commitments from other teammates, and pursue mission goals. Thus, we wish to guide its use by explicitly clarifying the scenarios for which it is *not* appropriate and highlighting problems that are outside our problem space.

Firstly, Hoplites is not designed for task-oriented problems such as task allocation and decomposition. Even though these problems require planning (e.g. to determine which robot is suitable for a task) and possibly coupling between tasks (e.g. if tasks have precedence constraints or require multiple robots), they ultimately ask the question "Which robots should complete which mission components?" Solving these problems does not require tight coordination of actions during execution and so they are not part of our problem space. In contrast, Hoplites answers the question, "*How* should robots complete their mission components?" and which often does require close interactions between robots during execution. Although one could formulate Hoplites to solve some task-oriented problems, passive and active coordination are unlikely perform as well as approaches such as Zlot's

task trees [11] which have been shown to solve very complex instances of these problems.

Secondly, successful tight coordination often requires significant information sharing between teammates because they must have knowledge of each other's activities to successfully choose actions. Some approaches utilize implicit communication through observations of teammates' state [14, 27] to facilitate tight coordination. Hoplites, on the other hand, exploits explicit communication to provide many of the improvements it has over other approaches. Moreover, Hoplites is robust to communication failure in that it will gracefully degrade into passive coordination or P-MVERT (discussed in Chapter 6). Nevertheless, if it is known in advance that explicit communication is limited or unavailable, it is more efficient in terms of implementation to utilize simpler approaches that depend only upon implicit communication.

Thirdly, planning and negotiation between teammates requires time. In highly dynamic domains (e.g. robot soccer [25]) plans can become invalid faster than robots can execute them, so the effort of actively coordinating will be wasted and may in practice slow the system down. Indeed, this is true of any approach that expends significant resources to make long-term decisions. Thus, in such environments we recommend simpler, more responsive techniques such as reactive or behavior-based approaches.

Lastly, as we described in Section 1.2, Hoplites in particular and distributed approaches in general may not be suitable for some problems that require both tight coordination and long-term planning but for which local management of constraints is not possible. Because these problems cannot be decoupled into local components, the benefits usually associated with distributed approaches are difficult to harness and centralized approaches may be preferable.

1.6 Summary of contributions

This dissertation makes several contributions to the multirobot literature which we briefly mention here and discuss in greater detail in Chapter 8:

A general and adaptive approach to addressing our problem space. Hoplites is a market-based coordination framework in which the complexity and strength of the coordination adapt to the difficulty of the problem. This adaptiveness enables Hoplites to produce better solutions than both distributed and centralized approaches while remaining computationally competitive with distributed approaches. Additionally, Hoplites is widely applicable to real-world problems because it is general, computationally feasible, scalable, operates under uncertainty, and improves solutions with new information.

Formalization of the problem space. This dissertation is the first to identify the common properties of problems in our problem space and formalize them alongside other multirobot problems. In doing so, it clarifies problem requirements and allows us to share solutions between problems that may have previously appeared disparate.

First application of market-based approaches to tight coordination. Until now, market-based frameworks have only been applied to loosely-coordinated teams. Hoplites extends the application range of these approaches by enabling robots to buy and sell tightly coupled plans; in doing so, it paves the way for future work.

First evaluation of and recommendations for planning algorithms for tight coordination. This dissertation is the first to outline a set of general planning approaches that solve different problem scenarios in the problem space. These techniques range from simple to complex and diversely negotiate the tradeoff between speed and completeness. This dissertation also provides guidelines on selecting and combining a set of these techniques into a single planning toolbox.

Significant improvement of previous state-of-the-art coordination framework. This dissertation improves the previous state-of-the-art coordination framework, MVERT[14] which experiments show results in a five-fold increase in performance.

1.7 Organization

The remainder of this thesis describes the Hoplites approach in detail and presents results from simulation and real robot experiments.

Chapter 2 describes related work in multirobot coordination in more detail. We discuss the full range of distributed to centralized coordination approaches and their applicability to our problem space. We find that current real-world approaches that enable tight-coordination use hill-climbing techniques that do not allow planning. Conversely, approaches that employ planning are limited to tasks whose components can be decoupled. Thus, the need remains for a coordination framework that achieves both planning and tight-coordination simultaneously.

Chapter 3 provides an introduction to market-based approaches to coordination, their benefits and drawbacks, and their applicability to our problem space. Although they are not designed for tight-coordination and although the challenges of loose and tight coordination are very different, we believe that market-based approaches can be extended to enable

tightly-coupled planning during execution. This extension is one of the contributions of this thesis.

Chapter 4 presents the Hoplites framework. We provide intuition to the approach through several examples, describe how it extends market-based frameworks, and discuss active and passive coordination in detail. We also walk the reader through a number of important design considerations. Finally, we explain how Hoplites meets the requirements of our problem space and how it successfully performs under real-world conditions.

Chapter 5 describes general planning techniques for solving different aspects of the planning problem. We illustrate how these approaches can be incorporated into the Hoplites planning toolbox to offer robots maximum flexibility in choosing the most efficient planner for the current planning scenario.

Chapters 6 and 7 describes in detail our implementation of Hoplites for the security sweep and constrained exploration domains. We use these different domains to explore the range of framework features we described in Chapter 4. We also demonstrate Hoplites's ability to scale to large teams and operate in the presence of uncertainty. Finally, we compare Hoplites to competing approaches in each domain and provide field experiments using both small indoor robots and large outdoor vehicles.

Chapter 8 presents conclusions and highlights the major contributions of this thesis. We also describe future work for multirobot coordination in general and areas of exciting new research that have been created by this thesis.

1.8 Publication note

Parts of this thesis have appeared in previous publications. In Chapter 3, the illustrative examples and portions of the discussion were first introduced in a survey by Dias et. al. [37]. Discussions and results relating to constrained exploration and security sweep were also previously published in conference papers [38, 39].

Chapter 2

Related Work

In virtually all robotic application domains, 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 multirobot systems where the complexity of algorithms for computing optimal solutions can be exponential in the number of robots. In the general multirobot 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 review the general literature but 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 simultaneously. Lastly, we discuss related work in the multiagent literature.

2.1 Centralized approaches

Centralized approaches employ a single agent to plan for the entire team. This agent can simultaneously consider the entire environment and the interactions of all team members; thus, by encoding planned and tight coordination directly into the problem, it can in theory produce optimal solutions. In reality, optimally coordinating more than a few robots in this way quickly becomes intractable since the complexity of algorithms that plan joint actions is usually exponential in the number of robots. Thus, central control of robots is not feasible in real world scenarios. 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 can fail. Thus, they are not

robust to malfunction or failure of the planning agent.

For these reasons, truly centralized systems are rarely used for general tight coordination of a group of robots. Khatib et al. [16] 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. Similar work has been done by Osumi et al. [18] and Ota et al. [19]. Unfortunately, these approaches become exceedingly complex as we increase the number of robots. Additionally, they are specific coordination techniques for manipulation tasks and not easily applied to other domains.

Esposito and Dunbar [40] and Schouwenaars et al. [41] use centralized approaches to enable both planning and tight coordination for the constrained exploration problem. Both exploit the property that the team can maintain communication constraints if robots form a chain in which each robot maintains communication contact with one particular neighboring teammate. As we discuss in Chapter 7, these approaches are not general because the solution is only applicable to this particular formulation of this particular domain. They also suffer from single points of failure.

Bowling et al. [25] successfully use a centralized approach to coordinate a team of five small robots playing soccer. Drawbacks typically associated with centralized approaches are mitigated both by the domain and by the implementation. The complexities of planning are reduced by the incorporation of predefined "plays" which specify roles and actions of individual robots and which can be invoked quickly. The difficulty of transmitting new information is reduced in two ways: firstly, the environment (not including robots) is completely known prior to play, and, secondly, a very-fast overhead camera is used to capture the locations of the balls, the home team, and the opponent team.

Partially-centralized systems have been used in loosely coordinated teams for task allocation. Caloud et al. [42] centrally allocate a variety of tasks such as "go-to-location" and "retrieve-object" to robots that independently complete their tasks. Brummit and Stentz [43] use a centralized system to demonstrate the benefits of reallocating goal points for a group of robots visiting a set of locations. Koes et. al. [44] use a centralized Mixed Integer Linear Program (MILP) to allocate tasks that are temporally constrained and which may require multiple robots for each task. The problem of coordinating the paths of multiple robots to avoid collisions has also been addressed using partially-centralized systems [45, 46, 47, 48]. Generally, robot paths are generated using decoupled planners but conflicts in paths are resolved centrally.

2.2 Distributed approaches

Distributed approaches dominate current multirobot 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* interact via specific coordination protocols.

Emergent approaches to coordination.

Emergent coordination strategies are most common for facilitating tight coordination in multirobot teams. They are based on the idea that coordinated group activity can successfully emerge from a collection of simple but carefully-tailored individual activities. Accordingly, coordination in robot teams is a byproduct of robots following a set of rules that map sensor data and robot states to particular actions or behaviors. Robots can choose actions very quickly and with minimal computation since the rules are simple. Tight coordination is possible because a robot can respond quickly and in a prescribed manner to the actions of its teammates. By the same token, coordination is limited to interactions that can be simply expressed, making such approaches insufficient in domains that require advanced planning of complex interactions between robots. In general, such strategies are robust to failure and deal well with uncertainty, although this is less true in tightly-coordinated teams. Matarić [49] provides a more detailed discussion of the principles behind emergent approaches to coordination. We consider two emergent strategies that differ in the complexity of action rules.

Reactive approaches

Reactive approaches are emergent coordination strategies in which the 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 [50] employ two robots to carry 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 a fast reactive loop. Similar work has been done by Hara et al. [21] and Kosuge et al. [20]. Pereira et. al. [51] use potential fields to guide robots to their goals while maintaining communication constraints. This approach depends upon robots' goals being in close proximity and results in robots taking paths in the same homotopic class. This body of work demonstrates that robots can successfully complete tasks 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 or deal with uncertainty and failures.

Behavior-based approaches

In *behavior-based* approaches, the rules are more complex and expressive. 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, and they can contribute to several tasks at the same time by executing multiple behaviors in parallel. The primary challenge for researchers developing behavior-based approaches is to create effective behavior selection strategies that allow robots to accomplish complex tasks without complex planning.

Behavior-based approaches have been used primarily in loosely coordinated teams engaged in tasks including exploration and mapping [10], emergency handling [12], hazardous clean-up [13, 52], and object tracking [14]. Less often, they are used to tightly coordinate teams tasked with carrying objects, moving in formation or exploring under communication constraints; we discuss this research in detail.

Much of the work dealing with behavior-based approaches to object transportation is more accurately described as partially reactive and partially behavior-based. Examples include the CAMPOUT architecture [5] and the BeRoSH system [22] in which robots use behaviors to coordinate different stages of the transport such as "assume formation," "approach target," and "deploy." The transport activity itself is accomplished using reactive coordination.

The CAMPOUT architecture has also been used for cooperative cliff descent [28] in which one robot traverses the face of a cliff while its teammates support it from their positions on the cliff ledge. Again, behaviors such as "maintain tension" and "match velocity" are used to coordinate phases of descent, while each behavior involves reactive interactions between teammates.

In formation tasks, robots must move as a group from one location to another while maintaining specific distances and orientations from one another. Robots must tightly coordinate to satisfy the constraints of the formation as they move. Behavior-based approaches are prevalent for maintaining robot formations because the constraints between the robots are simple and explicitly defined [53, 54, 55, 56, 57]. For brevity, we use the work by Balch and Hybinette [56] 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, and robots are also attracted to goals but repelled from obstacles. 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. [7] 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 but continues to provide connectivity. This task requires 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. This approach works well for a single robot, but it quickly becomes inadequate when we add more exploring robots and increase environmental complexity.

Wagner and Arkin [6] use a behavior-based approach for a similar communication-sensitive reconnaissance task: robots explore areas while trying to maintain network connectivity with each other. The philosophy behind this 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 robot controller by a centralized plan controller. This plan controller achieves 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, the plan controller can also assign different roles to different robots. This approach can suffer from a single point of failure (the plan controller) and is additionally not adaptable to arbitrary and complex instances of the problem.

Of all the prior work, we believe that Stroupe's MVERT [14] framework is the strongest candidate for accomplishing tasks like security sweep and constrained exploration. MVERT is a behavior-based framework in which robots choose their next actions based on the expected next actions of their teammates. Thus, MVERT facilitates planned coordination for very small time-steps. Like most behavior-based approaches, MVERT is fast, fault tolerant, and can operate under uncertainty. It also has virtually no communication needs and can also be applied to non-decomposable tasks. For these reasons we use MVERT for comparison with Hoplites. As we discuss in Chapter 6, however, MVERT's short lookahead traps it in local minima and renders it insufficient for complex problems.

Intentional approaches to coordination.

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 may be able to plan their future interactions. Here, we discuss purely intentional coordination strategies which have been almost exclusively used to decompose

and allocate tasks in loosely or moderately coordinated teams and not to tightly coordinate robots. We later discuss hybrid approaches which combine both intentional and emergent coordination strategies.

The collision avoidance problem we described in Section 2.1 has also been addressed using a distributed system with intentional coordination. In work by Azarm et al. [58], each robot plans its own path through the 2D workspace. As robots execute their paths, they detect impending collisions, and those robots involved in a local conflict negotiate to replan and adopt the lowest-cost paths. This approach works well for loosely coordinated teams where interactions are brief and sporadic.

In work by Dias [59] and Zlot et al. [9], robots use a *market-based approach* to perform loosely-coordinated exploration. That is, robots simulate a market economy in which they bid in auctions for the opportunity to complete tasks such as "explore-location-xy." Zlot and Stentz [11] also use a market-based approach to perform simultaneous task decomposition and allocation for abstract tasks. As shown by Dias et. al. [37], auctions provide a distributed, fault tolerant way for robots to decompose, allocate, exchange, and subcontract tasks that can be completed asynchronously and independently by individual robots or subgroups of robots.

Botelho and Alami [32, 33] present M+, a negotiation protocol for the allocation of tasks that can be completed by individual robots. These tasks are also partially ordered and thus require a moderately coordinated team. They achieve the additional coordination required to meet temporal constraints simply: robots broadcast the start and end of tasks, thereby indicating when other tasks can be started. This work is demonstrated in simulation in the context of hospital maintenance tasks and load transfer tasks.

Gerkey and Matarić also use a market-based approach which they call "MURDOCH" [60] to allocate object tracking, sentry duty, cleanup, and monitoring tasks to a loosely-coordinated team. They also address the problem of box pushing in which one "watcher" robot directs its teammates to push a box to the goal. The watcher observes the current position of the box, computes tasks such as "push-right-side-of-box" and "push-left-side-of-box" that will bring the box closer to the goal, and auctions these tasks to its two "pusher" teammates. The two 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 we mean 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 more similar to tasks in 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. [35] and MacKenzie [34] also use a market-based approach to coordinate vehicles completing temporally constrained subtasks. In both, a master task such as "destroy target" may require the completion of partially-ordered, independent child tasks such as "verify target," "bomb target," and "assess damage." The master task is allocated to a master robot who is responsible for the task. Then, the master robot auctions off the child tasks to other robots, sets deadline constraints, and monitors progress. Their approaches deal with much tighter task allocation and scheduling problems and facilitate closer intentional coordination between robots than the previous systems. However, because both also require the mission to be defined in terms of discrete subtasks, they cannot be readily used in domains such as object transportation that require tight coordination and may not permit decomposition.

Recent work by Gerkey et al. [30] is particularly relevant to Hoplites. They have developed a stochastic hill-climbing approach to planning for very small, tightly coordinated teams, which they call "Parish". In Parish, each robot plans for itself and increasingly many teammates up to some predetermined maximum size. Then each robot executes a probabilistically selected plan from the available set of such plans. This approach depends upon teammates having shared data structures and perfect information about the environment and each other. Such centralized planning for small teams is analogous to the use of the team plan in Hoplites which we discuss in Section 5 and can be thought of as a planning component of our framework. However, unlike Hoplites, Parish is not scalable to large teams because having perfect information about teammates and sharing such sizable data structures in large teams is not realistic to the demands of many application domains.

Lastly, Emery-Montemerlo [61] uses an intentional approach to provide tight coordination for a very limited set of problems. In this work, tight coordination problems are modeled as partially observable stochastic games (POSGs) and are solved using a series of distributed but linked Bayesian games (BaGAs). While this approach makes POSGs tractable and offers a powerful way to model decision making, it does not scale to large

teams or realistic problems in which the environment and the team problem are complex.

2.3 Hybrid approaches to coordination.

As shown by Balch and Hybinette [56] and Nguyen et al. [7], 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 intentionally coordinate to determine their relative roles when performing a task that requires tight coordination, but they actually accomplish the tight coordination using behavior-based or reactive approaches.

Chaimowicz et al. [15, 62] 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.

Lin and Hsu [24] and Jennings, Whelan, and Evans [23] both use a hybrid strategy to coordinate a distributed team of robots tasked with moving objects, some of which require at least two robots for transport. Both groups use an intentional approach to draw robots to obstacles that require transport, but they use an emergent approach to actually move the obstacles.

Vail and Veloso [26] use a hybrid approach to coordinate four Sony AIBOs playing soccer in the Robocup domain. Robots can take on roles such as “primary attacker,” “supporting attacker,” “supporting defender,” and “goalie.” With the exception of the goalie, the robots negotiate amongst themselves for these roles. Once these roles are assigned, they use shared potential fields to coordinate their activity.

In work by Naffin and Sukhatme [27], robots negotiate to organize and move in formation. Robots begin as singletons and, as they encounter each other, they 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 [56]. 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. [15], the constraints between robots are simple and explicitly defined in the task description.

Simmons et al. [63] propose a framework for more 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, in turn, 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. Their approach is mainly applicable to tasks that have two features. Firstly, the tasks must be decomposable into a distinct set of partially-ordered subtasks. Secondly, coordination between robots working on a subtask may be tight, but must be simple. That is, the interactions between teammates must be facilitated by reactive or behavior-based approaches and cannot require planning at the lowest levels. The tasks we hope to accomplish are typically not decomposable and they cannot be accomplished by simple interactions between teammates.

Jones et al. [64] describe a hybrid approach to dynamically- formed heterogeneous teams in which members perform tightly coupled tasks. They use a market-based approach to perform task allocation and use plays to tightly coordinate team members completing a task. A play consists of a set of predefined roles for each teammate engaged in the play and each role consists of a set of predefined actions. In this system, a task is announced in an auction call, and this task may require the coordination of several robots (for example if it requires multiple functionalities). A robot will compare this task against its playbook which contains several plays and find one that is suitable to the task. If such a play is

found, it holds an auction to allocate the remaining roles in the play. If all other roles are filled through this auction, the robot then bids on the original task. Assuming the task is awarded to the robot, this robot will coordinate the actions of the other team members involved in the play as long as the task is incomplete and the play is still relevant to the task. Individual robots will then execute their roles in the play and coordinate with their teammates to ensure the actions are executed in lock-step where required. This system is one of the first to address the complete multirobot coordination problem involving task decomposition, task allocation, task scheduling, and execution, and we discuss it further in Chapter 8. Moreover, it provides both tight coordination and planning through through the plays which can consist of long-term coupled actions between robots.

Nevertheless, several features prevent it from solving general domains in our problem space. First, like the work by Lemaire et al. [35] and MacKenzie [34], it requires the mission to consist of a set of distinct allocatable subtasks, which is not the case for problems such as security sweep where robots have a single, continuous mission with no subtasks. Secondly, as with Wagner and Arkin's [6] approach, it assumes that solutions can be described by a finite set of predefined plans. This is not the case for difficult problems such as constrained exploration which require planning in arbitrary and continuous environments and which further require replanning during execution. Thus, the notion of roles cannot be used to express the activities of robots. In sum, this work is well suited to allocating tasks which may require tight coordination between several robots, but the tasks that can be achieved by the plays method are limited to those that can be described by predefined interactions between the teammates. This set of tasks does not include constrained exploration and security sweep.

2.4 Multiagent coordination

Multiagent teams are related to multirobot teams in a number of ways: they solve similar problems (e.g. task allocation and constraint satisfaction), face similar challenges (e.g. limited information and failures), and achieve coordination using similar approaches (e.g. auctions). Nevertheless, multirobot teams are embodied and physically interact with their environment while purely multiagent teams are virtual. This means that while robots often operate in dangerous environments (e.g. flooded mines), agents are relatively "safe." Moreover, robots have significantly more uncertainty since their local state often depends on the state of the environment, which can be very difficult to assess.

This fundamental difference of physical embodiment has significant implications for our problem space. Namely, the problem of tight coordination throughout execution is almost nonexistent in multiagent systems. For example, there is no multiagent task analogous to the multirobot task of moving a piano using three robots. Nor is there a task analogous

to a team of robots traversing an environment while maintaining line of sight connectivity. Certainly there are domains that require scheduling of tasks but these involves solving significantly different coordination problems.

2.5 Summary

In summary, we find that current approaches to multirobot coordination that enable tight-coordination use short-term planning techniques to remain tractable. Conversely, approaches that employ long-term planning to complete tasks are limited to tasks whose components can be decoupled. The few approaches that provide both features are centralized; as such, they are unresponsive, do not scale to larger teams, and suffer from single points of failure. Consequently, current approaches to multirobot coordination do not address the needs of the class of tasks that require both extensive planning and tight coordination.

Chapter 3

Market-Based Multirobot Coordination

In this chapter we introduce market-based approaches to multirobot coordination. We first provide motivation and intuition for the approach and present an illustrative example. We then discuss the benefits of market-based approaches and how existing market-based approaches relate to our problem space. Dias et. al. [37] provide a more comprehensive overview of market-based approaches and also survey the literature in the field.

3.1 Overview

Humans have met the challenges of coordination for thousands of years with increasingly sophisticated market economies. In these economies, self-interested individuals and groups trade goods and services to maximize their own profit; simultaneously, this redistribution results in an efficient production of output for the system as a whole. Researchers have recently applied the principles of market economies to multirobot coordination. In market-based multirobot systems, robots are designed as self-interested agents that operate in a virtual economy and buy and sell tasks and resources to maximize their own profit; this simultaneously produces efficient solutions to the team's mission.

Most often, markets are used to solve the task allocation problem of assigning tasks to team members so that the overall mission is completed efficiently. Both the tasks that must be completed and the resources available to complete these tasks are treated as commodities of measurable worth that can be traded. Each task a robot completes generates some revenue for the robot 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; the greater the contribution of the task to the team's goal, the greater the revenue. The cost function, $C : R \rightarrow \Re^+$, is a mapping that reflects the quantity of resources

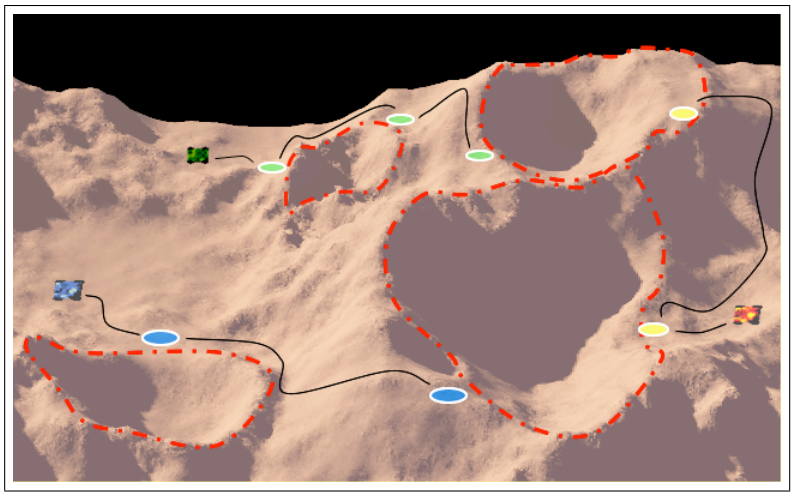


Figure 3.1: An illustration of three robots exploring Mars. The robots' task is to gather data around the four craters (outlined by red dashed lines), which can be achieved by visiting the target sites (represented by colored ovals).

the robot consumes, such as fuel or network bandwidth. An allocation is achieved by robots competing in auctions to win tasks that will generate the greatest profit for themselves, defined as revenue minus cost. The idea is that, through auctions and negotiations, tasks will be awarded to the robots best able to complete them and thus produce efficient team solutions.

3.2 A simple example

To illustrate this market-based approach more concretely, consider a team of robots performing a distributed sensing mission on Mars. As illustrated in Figure 3.1, the robots must gather data from specific sites of interest to scientists while also minimizing their energy expenditure. One important aspect of completing the mission is to determine which robot should visit each site. We can solve this problem using a market-based approach in which robots compete in auctions for each task of visiting a site. After estimating their resource usage for an offered task and submitting bids based on those expected costs, the robot with the best bid is awarded a contract for that site.

Suppose that we offer a maximum reward of $50 for each task and that robots incur a cost of $2 for each meter of travel (since the resource of concern is energy consumed). This $50 is a *reserve price* that essentially says that the task should only be attempted if the site can be reached by increasing one's path length by less than 25 meters. Further suppose that a robot A is only 5 meters from a site S. Since A would have to spend $10 to complete the task, it bids $10. Meanwhile, a robot B that is 10 meters from the site bids $20. A is

awarded the contract because it can perform the task more efficiently and for less than the reserve price.

3.3 Benefits of market based approaches

Market-based approaches have a number of positive features that have made them popular coordination methods. Firstly, they require users to formalize their preferences for the different solutions through global utility and local profit functions. This facilitates the comparison of different approaches and different solutions. Second, market-based techniques can provide guarantees on solution quality in certain circumstances [65]. Third, market mechanisms such as auctions provide a concise and simple way to compare the importance of different tasks and robots' abilities to perform these tasks; thus they use communication and information efficiently.

Here we further discuss the features of computational efficiency, responsiveness, and generality that we highlighted in Chapter 1 as most important for this thesis.

Computational efficiency

Market-based approaches are often able to distribute the planning required for task allocation or mission decomposition through the auction process: each robot locally plans a solution to the offered tasks, computes its costs, and encapsulates the costs in its bids. This process is illustrated in the Mars exploration example: each robot determines its own cost of visiting different sites. By distributing this planning through the auction process, markets provide a simple method of making multirobot task allocation tractable. Additionally, Dias [59] has shown how market-based approaches can produce better solutions by opportunistically allowing "pockets" of centralized optimization to emerge within subgroups of the team when resources permit. In our running example, for instance, the team might begin with a suboptimal allocation of sites; this could have been caused by an originally inaccurate map of the environment which, in turn, resulted in inaccurate bids. At some point during execution (perhaps when map information is improved by sensor measurements), a robot might find a better distribution of sites for some subset of its teammates. It can purchase the tasks from the original holders and subcontract them to the new holders and pocket the cost difference as profit. Simultaneously, this results in a better team solution.

Thus, markets can dynamically adjust the computational energy spent and the quality of solutions produced based on the resources available and the needs of the task. This is a primary reason why we have developed our own approach as a market-based framework, and we significantly extend this feature to provide adaptive coordination.

Generality

Market based approaches are general because the principles of cost and revenue are applicable to almost any multirobot domain, just as they are applicable to almost any human coordination problem. Indeed, market-based approaches have already been applied to a wide range of applications, including exploration and mapping [66, 67, 63], box pushing [60], surveillance and reconnaissance [68], assembly and construction [17], soccer [69], and treasure hunt [70]. Nevertheless, to date they have not been used to provide tight coordination, in part because it is unclear how tight coordination problems can be modeled with cost and revenue. That is one of this contributions of this thesis.

Uncertainty and Responsiveness

As we mentioned in the introduction, real-world domains always involve some uncertainty in the environment, the team, and the task. A successful coordination framework must enable the team to further the mission without requiring complete or perfect information. Market-based approaches fulfill this requirement. Suppose that in our Mars example, little terrain information is available to the robots when the initial allocation of sites must be made. The robots can still compute their bids based on whatever limited knowledge they have and based on some assumptions about the environment (e.g. by assuming that unknown areas contain no obstacles). Although this may produce a suboptimal allocation, the mission can still be completed.

Moreover, market-based approaches are responsive to changes and can improve the team solution when new information is received. The most common approach is by allowing robots to auction new tasks and reauction existing tasks during execution. For example, as the robots on Mars begin to complete their mission, they will obtain new information about the terrain from their sensors. The robots can improve the current solution by reallocating tasks through further auctions [59, 71]. Similarly, when robots malfunction or fail, the tasks that they have been assigned can be reallocated to other members of the team [72, 60]. Finally, new tasks can be created and assigned by holding new auctions during execution [64, 60, 63]

3.4 Market-based approaches applied to the problem space

Market-based approaches are almost always used to facilitate loose coordination for a number of reasons. Firstly, market-mechanisms are well-equipped to distribute mission planning (e.g. through locally computed bids) and mission tasks (e.g. through auctions). Neither of these features are easily applied to tightly-coordinated missions in which planning requires consideration of multiple teammates and in which components cannot be easily distributed.

Secondly, as we described earlier, the interaction between tightly-coordinating teammates are simple in most domains that researchers have investigated and can be accomplished successfully by emergent coordination approaches. These approaches are less expensive in terms of design, computation, and communication than market-based approaches and are thus preferable in most cases of tight coordination. Thirdly, some tightly-coordinated missions require extremely fast responses to teammates' actions. For example if robots are jointly carrying a fragile object, a slip by one robot might require instantaneous compensation by its teammate to keep the object safe. Since market mechanisms typically require more time than this for bidding and negotiation, market-based approaches may be too slow to meet this high degree of responsiveness.

Thus, to date, the role of market-based approaches in tight-coordination has been extremely limited. At most, markets are used to assign roles or tasks that require tight coordination, but execution is achieved using simple approaches. For instance, Simmons *et al.* [17] use a heterogeneous team to perform large-scale construction. They propose a market-based approach to secure teammate participation in manipulation tasks that require tight coordination, but execute the tight coordination using a reactive approach.

Although emergent approaches have been sufficient in the past, the class of tasks we are interested in cannot be solved successfully using these approaches because they require extensive planning of teammates' interactions. As discussed earlier, market-based approaches are well-equipped to facilitate coordinated planning during mission decomposition and task allocation, but they do not facilitate planning of coordinated actions during execution. This is because almost all market-based approaches deal with loosely-coordinated teams in which tasks can be achieved by robots independently. Clearly, current methods of planning for task execution in market-based approaches are insufficient.

Despite that market-based approaches are not designed for tight-coordination and that the challenges of loose and tight coordination are so different, we believe that market-based approaches can be extended to enable tightly-coupled planning during execution. This extension is one of the contributions of this thesis, and by combining it with the other benefits of market-based approaches, we believe they can be very successful in solving missions in our problem space.

3.5 Summary

In this chapter we introduced market-based approaches to multirobot coordination. In these approaches, robots act as self-interested agents operating in a virtual economy where the team's tasks and resources are of monetary value. Robots buy and sell these commodities over the market to maximize their own wealth, and, in a properly designed system, this price-driven redistribution simultaneously results in good team solutions. Market-based

approaches have a number of desirable properties including computational efficiency, flexibility, robustness, and generality. To date, market-based approaches have only been used to loosely coordinate teammates. Despite this, we believe that they can be extended to our problem space; this extension is one of the major contributions of this thesis.

Chapter 4

The Hoplites Framework

In this chapter we present the Hoplites framework which is the primary contribution of this thesis. To explain many of the concepts in this section, we will use a running instance of the constrained exploration problem in which a team of robots must visit a set of target locations while ensuring line of sight team connectivity. For simplicity, we omit the requirement of contact with a base station included in earlier examples. An instance of this problem with two robots is illustrated in Figure 4.1 *(a)*. We abbreviate this problem to LOSEX, short for line of sight EXploration.

We begin in Section 4.1 by providing intuition for the framework in the context of LOSEX. In Sections 4.2 and 4.3, we describe key components of the framework including extensions to market-based frameworks and coordination protocols, and we discuss design considerations in Sections 4.5 and 4.6. We then show in Sections 4.7 and 4.8 how Hoplites satisfies both the coordination requirements and the real-world requirements of the domains in our problem space.

4.1 Intuition

To intuitively explain Hoplites we first discuss the features of our problem space and the properties we are trying to capture. We then concisely present the features of the Hoplites framework that enable it to solve problems in our space better than other approaches.

Problem space features

Although planning for tightly-coupled multirobot systems is a difficult problem, this difficulty can vary greatly between different instances of the same domain: it depends significantly on the complexity of the environment and the size of the team. In LOSEX, for example, a cluttered environment is more challenging than an obstacle-free environment that permits unobstructed visibility and thus full network connectivity. Additionally, the

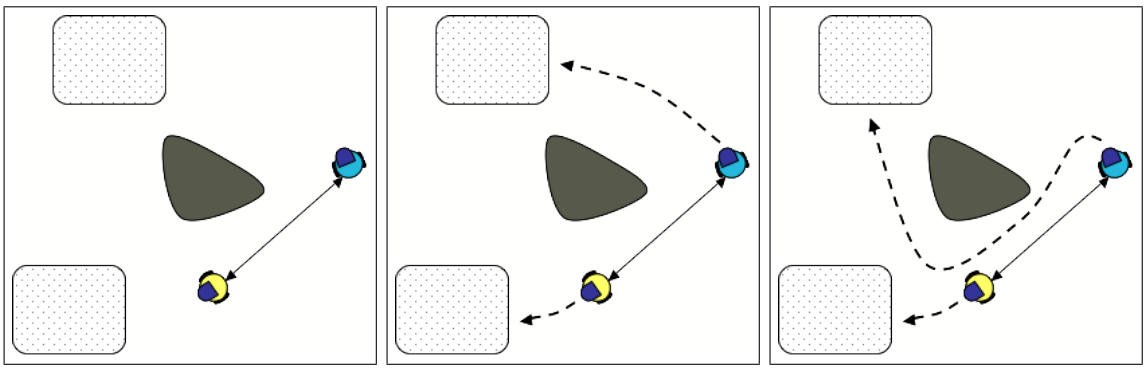


Figure 4.1: An example of LOSEX with two robots, two target regions (marked as dotted rectangles), and one obstacle (in dark grey)*(a)*. The shortest paths (marked by dashed lines) to the target regions would violate line of sight constraints *(b)*. The problem can be solved if one robot travels around the far side of the obstacle to arrive at its target region *(c)*.

same instance of a problem can be harder for a team of only five robots than for a team of fifty robots because the latter has greater visibility and thus greater connectivity. The difficulty of the task can also vary within a single instance of the domain. In environment with only one cluster of obstacles in a corner, the empty portions can be easily explored, but the cluttered corner may require close coordination between many teammates to ensure connectivity.

For efficiency, we want a coordination method that is only as complex as necessary to solve the problem at hand. That is, when robots are faced with an easy problem scenario, they should be able to coordinate with minimum communication and computation expense while still producing good solutions. Alternatively, in hard scenarios, robots should be equipped with a complex coordination method that may require more resource expenditure to produce a good solution. The challenge, however, is the complexity of the problem cannot be known in advance or determined by inspection, and it may change during the course of a mission.

Hoplites

Hoplites is a market-based coordination framework in which the complexity and strength of the coordination adapt to the difficulty of the problem. Robots begin with *passive coordination* which is light on communication and computation and allows teammates to iteratively respond to each other's actions without directly influencing them. When passive coordination traps robots in local minima, *active coordination* improves solutions by enabling robots to influence each other directly by purchasing each other's participation in complex

plans over the market. Active coordination consumes more resources but is also capable of producing much better solutions.

These features directly lead to the major contribution of this thesis: that Hoplites is best in class in solving problems in our space. Specifically, Hoplites outperforms competing distributed approaches by enabling pockets of complex coordination between several robots that result in better global solutions. Moreover, by selectively injecting this complexity, Hoplites provides these improvements while remaining computationally competitive with other distributed approaches. Finally, this selectivity also enables Hoplites to produce better solutions than centralized approaches which, for tractability, must resort to algorithms that produce feasible but very sub-optimal solutions. Hoplites, on the other hand, selectively uses these approaches and thus pays the price of significant sub-optimality only when necessary.

4.2 Hoplites as an extended market-based framework

We have designed Hoplites as a market-based framework to harness many of the associated advantages, including generality, speed, and responsiveness. We discuss these properties later Section 4.8.

As we explained in Chapter 3, most market-based approaches deal only with coordinating team members to allocate tasks or roles. Essentially, robots coordinate to answer the question "Which robots should complete which mission components?" Solving this means reasoning about mission at the level of its component tasks. For example, in our distributed data collection mission from Section 3.2, robots evaluate the costs of visiting some set of collection sites. The team is able to arrive at a solution by allowing each robot to independently determine how it would solve the different components and then using auctions to assign components to the most capable robots.

To solve the problems in our stated problem space, however, robots must additionally answer the question "*How* should robots complete their mission components?" Solving this means reasoning about the team mission at a much finer-granularity. For example, in constrained exploration, robots must determine not just which areas to assign to which robots, but also *how* these robots should explore them to ensure connectivity. This requires reasoning about every single action (and not just the larger task) and about the impact of teammates' actions on each other (and not just a robot's actions in isolation).

Extensions to Market-Based Approaches

Hoplites extends market-based approaches to multirobot coordination in two specific novel and general ways to meet the demands of our domains; we highlight these extensions here and discuss them in detail in the remainder of this chapter. First, in all prior work, a

robot's profit for completing a task depends only the revenue generated by the task and the cost of the resources consumed in doing so; it does not depend on the state of other robots. We introduce a new type of term called a constraint penalty that encodes the importance of maintaining constraints between teammates and that does depend on teammates' states. This is a general technique for encouraging implicit tight coordination between robots in a competitive environment. Moreover, unlike revenue and cost terms which represent the value of physical properties of our system, the constraint penalty term represents an abstract property that is neither consumed (as resources are) nor completed (as tasks are).

Second, in the current body of literature, market-based approaches have only been used to solve task decomposition and allocation problems by enabling robots to buy and sell tasks or roles through auctions. In contrast, we enable robots to buy and sell participation in fine-grained tightly-coupled plans which, in turn, represent constraints in the system. This provides a way for robots to influence how their teammates achieve their individual goals and thus facilitates planned tight coordination to satisfy constraints. Moreover, these commodities are not easily incorporated into existing systems: they can neither be modeled as roles because they involve very detailed actions tailored to specific teammates, nor can they be modeled as tasks since they are input into the problem or produced during task decomposition.

Coupled cost and revenue.

In other market-based approaches, determining the revenue a robot receives and the cost it incurs requires only knowing the task it has completed and the amount of resources it consumed. In general, its teammates' states are irrelevant. However, to promote coordination in Hoplites, cost and revenue functions are coupled and do depend on teammates' states. For example, consider the instance of LOSEX in Figure 4.1 *(a)* where two robots must explore the two shaded regions of an environment containing a single large obstacle that obstructs line of sight. In this example, each robot receives revenue for each unit of area it discovers, incurs a cost for each meter it travels, *and* incurs a constraint penalty for each unit of time that it is without communication contact with its teammate. Since this penalty affects the profitability of the robots' actions, it forces each to consider its teammates' actions when choosing its own. In Figure 4.1 *(b)*, each robot considers traversing the shortest path to visit an unexplored region. While this path has high revenue for discovery and low cost for distance, it will be worse overall because it violates line of sight constraints. Instead, the longer path around the obstacle (as shown in Figure 4.1 *(c)*) is more profitable than the direct path because it maintains line of sight between the teammates.

Negotiating action-level plans.

Since most market-based approaches deal only with task allocation, only tasks can be bought and sold over the market and robots only coordinate in the task space. In our problem domains, robots must coordinate at a finer granularity in the action space to satisfy the team constraints. Thus, existing approaches are insufficient. We have extended market-based approaches to allow robots to buy and sell fine-grained, action-level plans. This enables robots to negotiate for the first time *how* tasks should be completed even after they have been allocated to a particular robot. We discuss this extension further in Section 4.3.

4.3 Coordination components

Let us return to the challenge of providing robots with a flexible coordination framework that dynamically adjusts the coordination complexity to match the task complexity. To this end, Hoplites consists of two coordination methods for easier and harder problem scenarios.

Passive coordination

Passive coordination is the simple, less expensive method employed by robots as a"base" strategy. Throughout execution, a robot's goal is to select plans that are most profitable given its teammates' simultaneous actions. In passive coordination, a robot first generates a set $P_{candidate}$ of plans that are possible in the environment. It then estimates the profitability of each plan $p \in P_{candidate}$ in the context of its teammates' actions and the environment using a profit function that combines revenue, costs, and penalties. Once the robot has chosen its most profitable plan p_{max}, it broadcasts p_{max} to its teammates. Its teammates use this information to reevaluate the expected profitability of their current plans, update these plans, and broadcast any changes back. In this way robots' planning cycles iteratively incorporate the most recent information about teammates' intentions. Consider the example in Figure 4.2 *(a)* where the robots r_0 and r_1 have chosen to explore regions 1 and 2, respectively. In this simple environment, each robot initially selects the shortest path to its target region as its best path p_{max}. When the robots receive each other's intended trajectories, they find that these are still the best paths. Thus, in one step, the team converges to the optimal solution using only passive coordination. As we demonstrate in Section 6, passive coordination alone can be effective in many environments where the best actions are easily discovered (e.g. in an uncluttered environment).

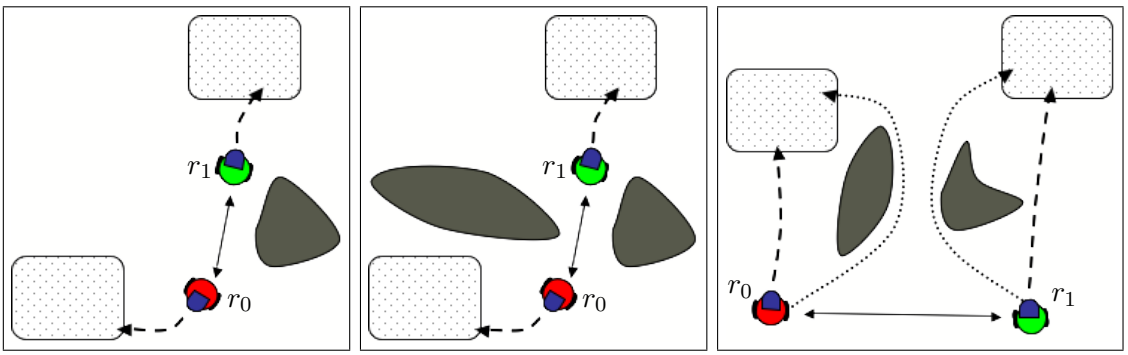


Figure 4.2: Robots r_0 and r_1 would like to explore the shaded regions of interest. In a simple environment they are successful using passive coordination *(a)* but become deadlocked in a slightly more complex environment *(b)*. In a third environment *(c)*, the two can reach their target regions in a much better way if they *both* travel through the valley created by the obstacles, but this solution will be much worse if only one travels through the valley.

Active coordination

In difficult environments, a good solution may require that teammates simultaneously perform complex coordinated actions. For two main reasons, however, these actions may not be produced using passive coordination alone.

First, market-based approaches attempt to produce cooperation through selfish decision making. However, the best solutions in complex scenarios often require some robots to perform actions that are more costly to them but are beneficial for the team overall. In passive coordination, where each robot chooses its most profitable option given its teammates actions, robots have no motivation to take these personally-detrimental actions, and the high-quality solutions will be passed up for poor solutions. For example, in Figure 4.2 *(b)*, an obstacle has been added to the environment that makes it more challenging for r_0 and r_1 to explore their target regions. To explore successfully, each robot now requires the assistance of its teammate to provide connectivity. However, using passive coordination, none of r_1's paths that would help r_0 reach its goal appear profitable to it: they involve cost expenditure for travel but do not garner any revenue from exploration. The same is true for r_0's paths that help r_1. In such situations, passive coordination traps the team in local minima because the global benefit of actions is not translated to individual profit.

Second, complex actions are also often *high-risk*. That is, if executed by all teammates, the actions will be immediately profitable, but, with only partial participation, the actions can be unprofitable. In contrast, *low-risk* actions will reap some rewards for a robot independent of it teammates' actions. When passively coordinating, a robot cannot guarantee its teammates' actions, and, without a commitment strategy, robots pass up high-risk ac-

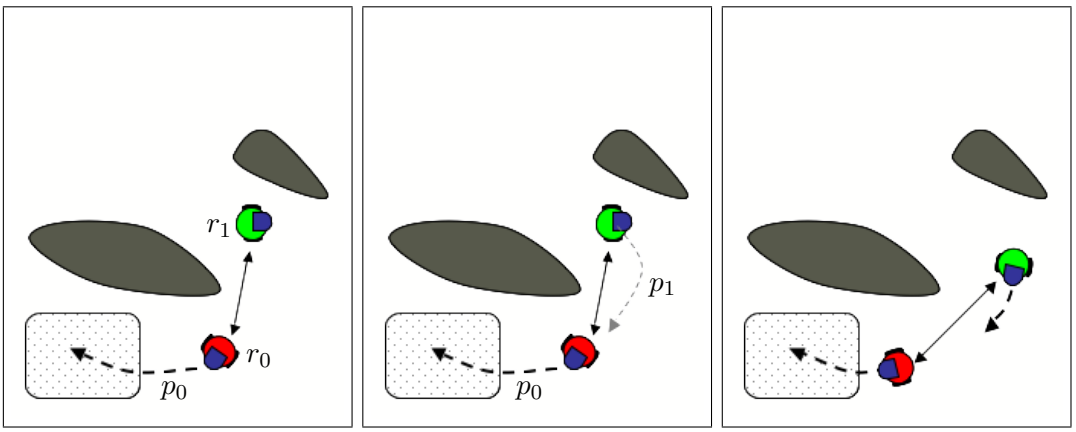
tions for low-risk ones that are less profitable and poorer solutions. For example, in a different environment in Figure 4.2 *(c)*, r_0 and r_1 again each have a region to explore. The best solution is for both robots to travel in the valley created by the two obstacles to maintain connectivity, even though this path is longer. However, if only one of the two goes through the valley, it will be more costly than if it had traveled directly to its target region because it requires greater expenditure in energy but does not provide connectivity. Without being able to ensure that their teammate would travel through the valley, they may both hedge their bets and opt for the poor but safe solution of traveling directly to their targets.

The active coordination mechanism can be used to escape both types of local minima. In a properly designed market-based system, if the team solution is poor, then it is also unprofitable for at least some members of the team. Active coordination exploits this property: it enables a robot to pay its teammates for taking actions that are inherently profitable to it but not necessarily directly profitable to them. By redistributing wealth, actions that are beneficial to the team can now also be profitable to the individual robots, and this makes cooperation a profitable rather than an altruistic endeavor. Furthermore, when a robot pays its teammates, those teammates are obligated to take the actions requested. Thus, teammates can guarantee each other's activities and take high-risk, high-profit actions. Active coordination is related to the idea of leaders and opportunistic optimization developed by Dias and Stentz [73] for loosely-coordinated teams controlled by market-based approaches.

4.4 An illustrative example

To illustrate these coordination mechanisms, let us consider the scenario in Figure 4.3 *(a)* in which two robots r_0 and r_1 are solving LOSEX in an environment with one target region and two obstacles. Let us assume that there is a high penalty for losing line of sight compared to the energy cost of moving around the environment. Here, r_1 has chosen not to explore the target region because it is too far away; instead it remains stationary. Meanwhile, r_0 attempts to explore the region using passive coordination, but the least-cost solution p_0 that achieves this is unprofitable because the revenue from exploring the target does not cover the penalty cost of losing contact with r_1. With passive coordination alone, r_0's most profitable plan p_{max} would be to do nothing and ignore the target region; p_{max} would have a profit of 0 since it neither costs anything nor achieves anything. Thus the team would be trapped in a local minima. This is similar to the example in Figure 4.2 *(b)*.

The active coordination mechanism produces the optimal solution. During planning, suppose r_0 finds that it will incur a minimum penalty of c_0 for taking p_0 because of the break in visual contact with r_1. It searches for a team plan P_{01} that would decrease this



(a) (b) (c)

Figure 4.3: In this example of LOSEX, r_0 robot would like to explore the shaded region but cannot using passive coordination without losing line of sight contact with r_1 *(a)*. By developing a team plan P_{01} with r_1 as shown in in grey in *(b)* and actively coordinating, however, r_0 can successfully reach its goal and also produce the optimal team solution *(c)*.

cost; it finds $P_{01} = \{p_0, p_1\}$ as shown in Figure 4.3 *(b)*. r_0 then requests a price quote q_1 from r_1 for P_{01}. This quote reflects the additional cost c_1 to r_1 for taking the proposed path p_1 instead of remaining stationary as it had intended. We expect that since the penalty for line of sight violations is higher than the cost of consuming resources, c_1 will be less than c_0 and r_0 will have a profit margin $m = c_0 - q_1 = c_0 - c_1$.

Clearly, this solution results in a positive profit for r_0 even after it pays off r_1. However, m must also be greater than r_0's original best alternative, p_{max}, which ultimately set's the maximum price that r_0 would pay its teammates for their assistance. This value is known as the *reserve* price. Here, m is greater than the profit of p_{max} (which was 0), so r_0 can then offer q_1 to r_1 to offset its costs and pocket m for itself. r_0 could further offer a portion of m to r_1 if r_1 had competing alternatives to P_{01}. Now, r_1 will find that it can do at least as well as it would have with its original plan of doing nothing, so it will accept the offer. In the pursuit of larger profits over the market, the team arrives at the optimal solution.

In this way, active coordination allows robots to explicitly weigh the benefits of certain actions (e.g. exploring the region) against the costs (e.g. the extra effort required by teammates). Furthermore, it enables robots to share the benefits with their teammates by paying them and thus promotes cooperation in a competitive environment. Also, it provides a contract mechanism by which robots can guarantee each other's actions. In sum, with active coordination, the team can escape local minima and solve the problem more efficiently.

Hoplites also enables the active coordination of several robots. In Figure 4.4*(a)*, we have

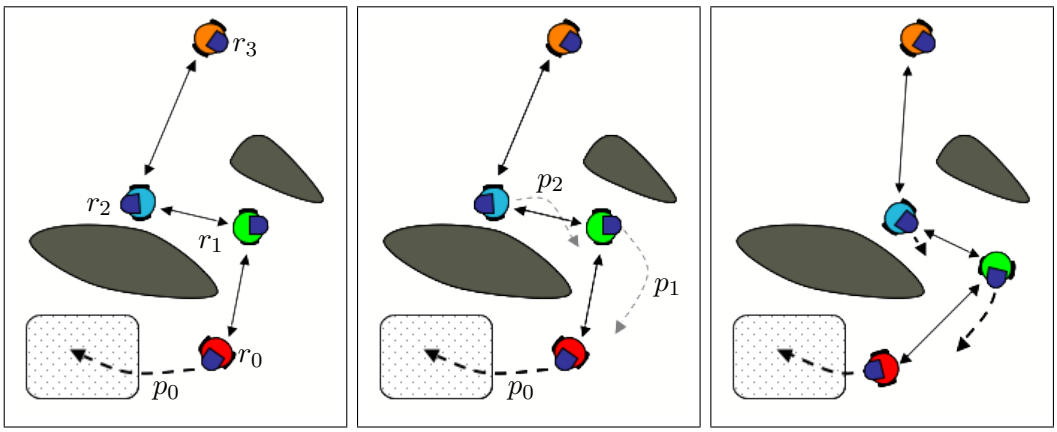


Figure 4.4: In this example of LOSEX, r_0 as teammates r_1, r_2, and r_3; it would still like to explore the shaded region without losing line of sight contact with r_1 *(a)*. Now, however, it must coordinate with both r_1 and r_2 *(b)* in order to successfully reach its goal and produce the optimal team solution *(c)*.

added two robots r_2 and r_3 to the team. Although this does not change the environment or r_0's goal of getting to the target region, it does mean that more robots must remain connected. As before, passive coordination will not enable r_0 to reach the region profitably and the team will again fall into a local minima.

If r_0 now proposes P_{01} to r_1, the price quote q_1 would be much higher than before because it will cause a line of sight violation between r_1 and r_2 and cause c_1, the added cost of taking p_1, to be much higher. Indeed, we can expect that the overall cost of taking p_1 will be greater than the cost c_0 because in addition to the line of sight penalty, c_1 includes the energy cost of taking p_1. So, $m = c_0 - c_1 < 0$ and r_0 will be unable to bid for r_1's participation.

There are two approaches that r_0 can take to solve the problem. The first is that r_0 can then develop a team plan P_{012} for all three teammates together. Here, $P_{012} = \{p_0, p_1, p_2\}$ as shown in Figure 4.4*(b)*. Now, the cost of r_1 taking p_1 when r_2 simultaneously takes p_2 is the same as it was in the previous example with only two robots. r_0 receives quotes q_1 and q_2, and we expect that since both of these costs only involve energy expenditures, they will be less than the cost of r_0's violations. That is, $m = c_0 - (q_1 + q_2) > 0$, and so r_0 can compensate its teammates, who will be able to do at least as well as they could have previously. The three robots will form a contract and thus the market will have found the optimal solution.

Nevertheless, in real systems, this approach of a single robot planning for many of its teammates ultimately will not scale for two main reasons. First, a single robot is unlikely

to have access to all the necessary information about all the teammates that is required to solve the problem. Indeed, the robot may not even know which teammates are required to solve the problem. Secondly, even with this information, a team plan for more than a few robots may not be computationally feasible. Fortunately, the unique elements of the Hoplites framework enable the team to solve the problem.

Suppose r_0 once again generates the plan P_{01} and r_1's quote is too high. Rather than quitting at this point, r_1 can continue to explore the problem. That is, r_1 can generate the team plan $P_{12} = \{p_1, p_2\}$ as shown in Figure 4.4 *(b)* and request a quote q_2 from r_2 that reflects the added cost c_2 to r_2 of taking p_2 instead of remaining stationary. We expect c_2 to be small since p_2 only adds energy cost (just as p_1 did in the previous example) and does not violate line of sight constraints with r_3. Thus, upon receiving q_2, r_1 can evaluate its new cost c_1' that is the cost of taking p_1 when r_2 simultaneously takes p_2. It can then send to r_0 a new price quote $q_1' = c_1' + q_2$. Again, we expect that $c_1' + c_2$ will be less than c_0, so r_0 will be have a profit margin $m = c_0 - q_1' = c_0 - (c_1' + c_2)$. r_0 will then offer q_1' to r_1 which, in turn, will offer q_2 to r_2. Both r_1 and r_2 find that they can do at least as well as they would have previously, so they will accept the respective offers. Here, the system is able to find the more complex optimal solution via the market.

As shown in this example, Hoplites makes it possible to efficiently coordinate several robots. Firstly, planning can occur through a series of linked plans (e.g. P_{01} and P_{12}), so a single agent does not have to plan for a large number of robots. Secondly, the utility of a plan (e.g. q_1 and q_2) is evaluated locally, so robots can use more accurate local information to generate more accurate cost estimates. Thirdly, by combining a series of price quotes (e.g. q_1'), robots can concisely transmit the costs and benefits of a collection of plans. This means that robots can investigate different solutions to the same problem; the most cost effective one is naturally adopted in the market. Thus, a robot (e.g. r_0) can benefit from and even cause and pay for the coordination between its teammates (e.g. r_1 and r_2) without ever being aware that the coordination occurs. In this way, the needs of one part of the team can be transmitted to and met by other parts of the team transparently and efficiently. As we demonstrate in Chapter 6, this chained coordination produces good solutions while consuming few computational and communication resources.

4.5 Practical considerations: problem setup

While these basic coordination methods round out the fundamentals of Hoplites, there are a number of practical considerations that must be addressed in order to implement the framework in practice. In this section we discuss the problem setup. Specifically, we look at how to translate an abstract domain description such as "explore an area while maintaining line of sight connectivity" into a concrete problem. In the next section, we discuss practical

Problem Setup

1. **Problem definition:** Identify problem goals, resources, and constraints. Define global constraints as a set of locally manageable constraints.
2. **Function Q:** Formally combine problem factors in a single function Q that measures the global solution quality. Define any weights necessary to trade off dissimilar factors.
3. **Function p:** Derive from Q the local profit function p that robots use to guide individual actions.

Coordination Setup

1. **Passive coordination:** Determine how and how often a robot should generate a set of candidate plans. Choose the set size, planning algorithm(s), and planning horizon.
2. **Switching mechanism:** Select a switching point between passive and active coordination, usually using a minimum profit threshold θ_{min}.
3. **Active coordination:**
   a) Teammates: Determine with which teammate(s) a robot should attempt actively coordination based on the local constraint structure.
   b) Team solutions: Determine how a team plan should be constructed and by which robot. Choose the algorithm(s) and planning horizon.
   c) Reserve path: Decide if and how a robots should compute a new reserve path to improve the best alternative option if active coordination fails.
   d) Market mechanism: Choose a market mechanism by which robots buy and sell plans (e.g. auctions, directed peer-to-peer sales)

Figure 4.5: An outline of the design considerations necessary to implement Hoplites. The first three relate to the problem setup and the second three relate to the coordination setup.

considerations for enabling passive and active coordination. We highlight all the steps one would use in designing a Hoplites solution in Figure 4.5 and fill in the steps for each of our implementations in Chapters 6 and 7.

Problem definition

We often begin with a very abstract idea of the problem we want to solve with a team of robots which we then need to precisely define for implementation. For our problem space, in particular, we must first restructure the original problem to make it feasible for a distributed system.

LOSEX, for example, is very similar to the constrained exploration problem we described

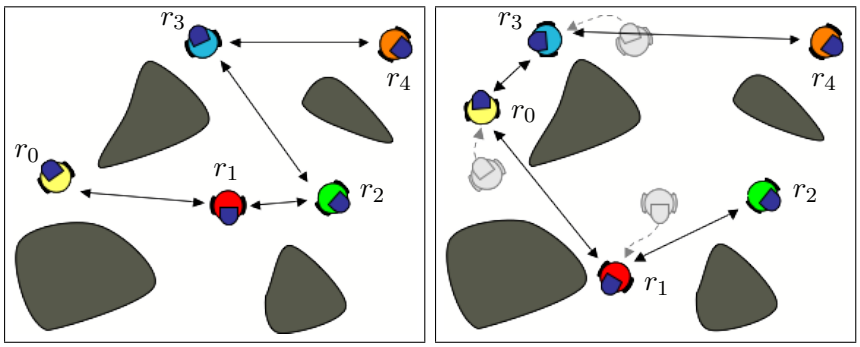


Figure 4.6: A team of five robots exploring an environment. Solid arrows indicate the line of sight communication links between teammates. *(Left)* Robot r_0 cannot explore the area without losing contact with r_1. *(Right)* By changing links, the team can restructure and provide r_0 with greater flexibility. The team members' previous positions are highlighted in light grey and their paths are shown as light grey arrows.

in Chapter 1 in that it is difficult to evaluate and intractable to solve if we describe it as a general connectivity problem. We can make it tractable, however, using an approach similar to our earlier method in Chapter 1. Each robot is assigned two specific neighboring teammates with which it must maintain line of sight connectivity (the two end robots have only one neighbor). This is illustrated in Figure 4.6 *(a)*, where the arrows connect neighboring teammates. With this formulation, our team of robots simulates a robotic arm with four prismatic and four revolute joints. The condition that all robots maintain contact with their designated teammates is sufficient (although not necessary) for maintaining team-wide constraints, and it is a much more tractable problem and more feasible to evaluate.

The disadvantage is that our restructuring method may be too conservative. For example in Figure 4.6 *(a)*, robot r_0 cannot move forward around the obstacle to its right because it must maintain contact with r_1, even though the team would still be connected if it did move. This is a general drawback of trying to piecewise locally capture global constraints, but it is necessary to make the problem feasible. We can overcome these difficulties by using a structure that meets the satisfiability condition but that is also minimally limiting. For instance, we might allow robots to swap their neighboring teammates as shown in Figure 4.6 *(b)* provided that the swap can guarantee satisfiability. Active coordination is particularly useful for this: r_0 can negotiate with r_3 and convince it to release its link with r_2 and adopt a link with r_0. We return to this flexible structure later in Chapter 7, but for now stay with the simple fixed-arm structure for simplicity.

There are many different ways to create a set of locally manageable constraints besides the fixed-arm structure. Others include bottom-up trees in which follower robots must maintain constraints with leader robots and top-down trees in which leader robots must

maintain constraints with follower robots. As we describe in Chapter 6, we use a fixed arm structure in security sweep because it closely maps to the original problem. In Chapter 7, we use a bottom-up tree structure for LOSEX that offers a good balance between maximizing flexibility and ensuring feasibility.

Finally, we also need to describe our goals and any resources of interest which we would like to conserve. Our mission in LOSEX is not just to maintain constraints, but to explore the environment. Additionally, in most missions, we are interested in conserving some resource of value (e.g. network bandwidth or fuel). Let us suppose that we are interested in minimizing the team's total energy consumption. Thus, we have defined our problem as maintaining a chain of line of sight constraints while maximizing the area explored and minimizing the energy consumed by the team.

Quantification of user preferences.

As with all market-based approaches (indeed, ideally with all multirobot approaches), we want to quantify the quality of a solution produced by the team so that solutions can be analyzed and compared. We must define a global utility function Q that combines all the factors that contribute to the quality of the solution. In LOSEX, we are concerned with maximizing both the area explored while minimizing the teams total energy consumption and the communication disconnections between designated pairs of teammates. One common implementation of Q is as a linear combination of the relevant factors:

$$Q = A\sum_{i=1}^{R} Area(r_i) - B\sum_{i=1}^{R} Energy(r_i) - \Gamma\sum_{t=0}^{T}\sum_{i=1}^{R} Disconnection(r_i, r_{i-1}, t) \qquad (4.1)$$

In this equation, R is the set of robots on the team, $Area(r_i)$ is the square units of new area explored by robot r_i over the course of the mission, $Energy(r_i)$ is the total amount of energy consumed by r_i during the mission, T is the duration of the mission, and $Disconnection(r_i, r_{i-1}, t)$ is a boolean indicating whether or not the constraint between r_i and r_{i-1} was maintained at time t. (Notice that we omit the disconnection between r_i and r_{i+1} since line of sight is symmetric, even though it is in our local formulation.) A, B, and Γ represent our relative preferences for each of these factors. For example, if the connectivity requirement in LOSEX is essential, we would have a high value of Γ relative to A and B. Alternatively, if connectivity is merely preferable, we might use a much lower value of Γ. Indeed, if connectivity is irrelevant, we can set Γ to zero and have a standard, unconstrained multirobot exploration problem.

Note that we are not always interested in the total expenditure of a resource; for example, we might want to minimize the mean energy expenditure or minimize the maximum amount

of energy used by any single robot. Furthermore, a linear combination is only one of many ways to express this relationship; we could instead use a quadratic combination, for example. Our preferences can also change over time; for instance, it may be more valuable to preserve energy at the beginning of an experiment than at the end, so B can increase as the mission continues. Clearly, the same solution can be evaluated in a number of different ways; thus the global utility function is valuable because it forces us to formalize both the problems we wish to solve and our preferences for different solutions.

Mapping global utility to local utility

Once we define our global utility function Q, we must then solve the credit assignment problem which asks, "What is the contribution of an individual agent's actions to the larger team objective?" In market-based approaches, this means deriving a profit function P that robots use to determine their rewards and costs for particular actions. A properly defined function will locally guide the robots towards maximizing Q.

In loosely coordinated teams, P can often be a perfect analogy to Q. For example, suppose we have the unconstrained multirobot exploration problem and the team is loosely coordinated; that is, Q is the same as in Equation 4.1 except that $\Gamma = 0$:

$$Q = A \sum_{i=1}^{R} Area(r_i) - B \sum_{i=1}^{R} Energy(r_i) \tag{4.2}$$

Then, the solution to the credit assignment problem is that each robot receives revenue for each unit of area that it was first to explored and incurs a cost for the energy it consumed during the mission. The resulting function P that determines a robot r_i's profit for an action a taken at time t that moves it to position p is

$$P(r_i, a, t, p) = \alpha\, Area(p, t) - \beta\, Energy(a) \tag{4.3}$$

Here, $area(p, t)$ is the square units of new area observed (it is time dependent because if any robot returns to p after it has already been visited, no new area will be credited to that robot). $energy(a)$ is the energy required to take action a and is not time or teammate dependent in this domain. We use α and β to represent the local weights; often it works well to use the same weights as in Q so that the relative importance of the different factors is consistent between the two functions. Note that we capitalize the weights when we are discussing them with respect to Q and leave them in lowercase when we are referring to P. Tovey et. al. [65] present many solutions to the credit assignment problem for a number of different global objective functions for loosely coordinated teams.

In the general case, credit assignment in tightly coordinated problems can be very challenging because robots' actions are interdependent and can have far-reaching consequences

for their teammates. Our problem space, however, has already forced us to reduce team-wide constraints to local constraints. While this original step was challenging, once it is solved, credit assignment can be straightforward. Each teammate is responsible only for maintaining the constraints with other teammates that were defined by the restructuring procedure. This results in a new profit function derived from Equation 4.1:

$$P(r_i, a, t, p) = \alpha\, Area(p, t) - \beta\, Energy(a) - \gamma\ (Disconn(r_i, r_{i-1}, t) + Disconn(r_i, r_{i+1}, t)) \tag{4.4}$$

Thus, a robot is penalized if the line of sight is broken between itself and one of its two neighbors. Furthermore, to evaluate the profitability of certain actions, the robot only needs to know the positions of those neighbors. Note that in this case, penalties are bi-directional: that is, two neighboring robots are both penalized for disruptions in their line of sight. However, uni-directional penalties, in which only one teammate is penalized, are also possible. Such penalties are particularly useful in tree structures: either the leader or the follower is responsible for maintaining a constraint but not both.

4.6 Practical considerations: active and passive coordination

Now that we have set up the problem structure and defined utility and profit functions, we must design the coordination mechanisms that will use these components.

Passive coordination design

The first step in Hoplites is for the robots to passively coordinate to solve the problem. We must decide how the robots should generate their candidate plans, how often they should re-generate these plans (both to extend the planning horizon and to take into account new information), and how often they should share these plans with their teammates.

Firstly, we want to strike a balance between generating many plans that largely explore the solution space while focusing our plan generation in ways that are likely to result in good solutions. Indeed, this is the challenge with many planning applications. The specific plan generation approach is very domain dependent. In domains where robots have a range of choices in their actions, for example if they are exploring an environment as much as possible without priority to any particular target areas, we find it is useful to cast a wider net. So, we might first choose a set of randomly generated goal points in the environment and then plan a randomized path to each of those points to get a very diverse sampling of the space. In contrast, when robots must achieve a particular goal, for example if a robot

must explore a particular region, we find that casting a wide net is often wasteful because few of those plans turn out to be viable. Instead, it is better to deterministically generate a few, high-quality plans that are most likely to achieve that goal.

Secondly, we must determine how far ahead to plan in time. This is called the *planning horizon*. The advantages of a longer planning horizon is that it allows robots to deal with problems well in advance and enables the team to find better solutions since the choice of immediate actions are influenced by their impact on later actions. The disadvantages are that a longer planning horizon requires more computational expenditure that, particularly in real-world environments with imperfect information, may be wasted as solutions turn out to be infeasible. The opposite is true of shorter planning horizons. We find that a good balance is achieved when robots plan out to the point where they are likely to have good information about the environment, provided this is within the bounds of their computational resources. For example, if a robot can observe the environment out to 50 meters, has a good prior map, and moves at a rate of 1 meter per second, then a planning horizon of about one minute might work very well. If, on the other hand, the robot is engaged in fast-paced game where the problem qualitatively changes every 2-3 seconds, a shorter planning horizon of a few seconds is more responsive and less wasteful.

Thirdly, we must decide how often the robots should replan. As with choosing a planning horizon, we want our replanning to respond to changes in the environment while not being wasteful. A simple and good strategy is to frequently reevaluate the profitability of the current plan (which is often very inexpensive) but to only replan (which is typically much more expenesive) when it becomes significantly less profitable.

Fourthly, robots must share information with each other. Once again, we must strike a balance between maximizing responsiveness and minimizing wasteful communication. We find that this balance is met when robots share only new or changing information. That is, robots only broadcast plans when they change and only share state information when it differs from its expected state.

With these features in place, the team is fully ready to employ passive coordination, which, as we've stated, successfully solves many simpler problems.

Choosing coordination methods

So far, we have equipped our robots with two coordination methods; we must still give them a way to choose between them. The general approach is for robots to switch from passive coordination to active coordination whenever their maximum profit for future actions drops below some threshold θ_{min}. This signals that the team's solution will also be poor. Ultimately, the choice of θ_{min} depends on the domain and the user's global objective. For example, if the connectivity requirement in LOSEX is absolutely essential, we would

prefer the robots to expend the extra resources required by active coordination. This could be achieved by having a high penalty for losing contact relative to the cost of energy and a high θ_{min}. Alternatively, if connectivity is just preferable, we might want the robots to switch only when disconnection will occur for a long period of time. Then, the penalty for losing contact would be lower relative to the cost of energy and θ_{min} would also be low.

As a second general rule, we want robots to expend time and communication resources to actively coordinate only when it is likely to result in a better, more profitable solution. Consider a robot whose future plans may all have a profit less than θ_{min}, for example because the amount of resources required by these plans is simply very high but no constraint violations occur. By the θ_{min} rule alone, this robot might try to actively coordinate to improve the profitability of its options. However, resource consumption such as energy or time for travel is typically independent of other robots' actions. For example, a robot's consumption of fuel when it traverses a path remains the same regardless of its teammates' simultaneous actions. Thus, using active coordination will almost certainly not help in these cases and the robot would end up wasting communication and computation. Constraint violations, on the other hand, are highly-dependent on teammates' actions and can often be avoided through active coordination. Thus, we recommend that a robot passively coordinates until it expects that its best plan will be less profitable than some θ_{min}, *specifically due to constraint violations*; then, it should actively coordinate with its teammates.

Choosing teammates

Once a robot decides that it should actively coordinate to improve its current prospects, it must choose the teammates with which this coordination must occur. This, too, is a very open, domain-dependent design decision. In LOSEX, for example, we might have a robot choose its n nearest teammates or the teammate that has been most successful in the past in keeping other team members connected. We might even use a ladder approach in which a robot starts with a simple option (e.g. its nearest n teammates) and, if that fails, tries increasingly complex options (e.g. its nearest $n+1$ teammates). This last approach is similar to the PARISH algorithm [30].

Although this may seem like an arbitrary decision at first, often it follows from the local constraint structure that we discussed in the previous section. For instance, if we use a linked-arm structure in LOSEX, then a robot r should first try to actively coordinate with its link neighbors. Firstly, these neighbors will have a direct incentive to cooperate with r if the constraints are bi-directional. Secondly, r will probably have more accurate information concerning these neighbors than about any other teammate and can thus reason about them much more accurately. Thirdly, and most importantly, these are the neighbors that actually have a direct impact on r's constraints. For these reasons, this is the approach

we use in both of our implementations. If coordinating with these neighbors fails, then the next attempt can be one of the many viable alternatives we have mentioned.

Planning team solutions

Once the teammates have been selected, a plan must be constructed that improves r's prospects. One of the major strengths of the Hoplites framework is that it is not limited to any particular planning approach; any algorithm can be utilized. This means that both traditional, all-purpose planners such as A* [74] can be used for their ease of implementation and consistent performance as well as domain-specific planners that have been tailored to the problem. Nevertheless, the choice of planning algorithm can significantly affect the computation requirements of generating plans and the quality of plans. We devote Chapter 5 entirely to discussing planning algorithms for our problem space.

We must still decide which robot should produce a solution plan for the subteam. There are two general approaches to this. We could either let the robot r that is attempting the active coordination also plan for the group, we could let members of the group propose solutions, or both. There are advantages and disadvantages to each of these methods; which one is preferable depends on the circumstances.

The advantage of r planing is that it is likely to have the most information about the constraint violations and its goals and therefore is best equipped to find a solution that meets its needs. The disadvantage is that r is unable to consider the constraints of its teammates and may not be well equipped to find a solution that also meets its teammates needs and is therefore less expensive. The alternative is for r to propose the actions it wishes to take and request candidate solutions from its teammates. This approach allows the teammates to explore many solutions while taking into account their own goals; thus, they are better able to find less expensive solutions. The disadvantage is that it is challenging to simultaneously coordinate several teammates in this manner. Finally, we could use both approaches by having r suggest a plan and also request alternative plans. Naturally, the advantage is that the robots can consider a larger number of solutions and increase the chances of finding a less costly one; the disadvantage is that this computation is itself complex and expensive.

In general we can break it down into a rough decision tree. If there are enough computation and communication resources, both approaches should be used. Otherwise, if the solution requires centrally coordinating the actions of more than one robot (excluding r), then r should plan because it can centrally coordinate its teammates. If only one robot needs to be coordinated with r, we then consider the range of solutions to r's request. If there is a limited number of ways in which the teammates could help r achieve its goal, then r should again plan because it has the most information about the goal and is still likely to hit upon the teammates' cheapest plans. Otherwise, if there are many ways in which

the teammate can satisfy the constraints, then the teammate should plan. These decision points (e.g. the range of teammates' solutions) are domain dependent and not clear-cut. Trial and error may be necessary to settle on an approach. Moreover, the approach need not be chosen in advance; it can be dynamically selected during execution based on the immediate resources available and the nature of the problem.

In security sweep, we use a bi-directional robot-arm structure for the team. This is both light on computation and communication since there are a limited number of solutions for both robots and they are easy to find; thus, we let both r and its teammate propose solutions. In LOSEX, there are a number of ways for teammates to assist r so we let them propose solutions.

On a final note, one's approach to passive coordination can and should affect one's approach to active coordination, and vice versa. For example, suppose that in passive coordination we use a very focussed approach to planning (perhaps because a simple solution usually works very well) that results in very limited set of candidate paths. Recall that a robot's best plan p_{max} from this solution set also creates the reserve price during active coordination that sets the maximum that the robot will pay for some team plan p_{team}. Then, if p_{max} comes from a very limited set of plans, it may be very poor and thus result in a high reserve price (i.e. the robot is willing to pay a lot for its teammates' cooperation because it does very poorly on its own). Indeed, the reserve price may be artificially poor: if our candidate set from passive coordination had been larger, a better solution might have been found than the current p_{max}, thus lowering the reserve price. Indeed a solution may have been found that would make active coordination unnecessary. We can compensate for this shortcoming during active coordination by having the robot generate a broader set of candidate solutions and select a new p_{max} from that set. This new p_{max} is likely to have a better reserve price (and thus a better solution) than the previous set. In this way, we can compensate for the simplicity of one coordination mechanism by adding complexity to the other.

The market mechanism

After a candidate team plan has been developed, teammates must go through a negotiation procedure to secure participation in a contract. Although most market-based approaches use auctions in which many teammates participate and bid, we believe that point-to-point negotiations are most effective in Hoplites. Auctions are efficient when the item for sale can be bought by many teammates because all the bids can be compared at once and the protocol is simple. In Hoplites, however, a robot often requests participation from very few of its teammates, for instance only its adjacent neighbors. Also, the plans for which the robot seeks participation are often tailored to particular teammates. In these scenarios, an

auction would be wasteful because it would cause robots that could never bid on those plans to still receive and consider the offers. We recommend and use point-to-point negotiations in which one robot makes a directed request to a particular teammate.

This mechanism is analogous to auctions with only one bidder. The "auctioneer" robot announces a plan that it would like participation in; this announcement is directed to one of its teammates. Then, that teammates return a bid that indicates the cost of its participation. The auctioneer "clears" the sale by comparing the bid to its reserve price. If the bid is less than the reserve price, a contract is made between the robots. Otherwise, the auctioneer must find an alternative solution.

4.7 Meeting coordination requirements

In the previous section we provided a general description of the coordination mechanisms in Hoplites. Here, we explicitly illustrate how Hoplites meets the three major requirements of our problem domains: tight coordination, planned coordination, and flexible coordination.

Tight coordination

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. The Hoplites framework is designed to facilitate tight coordination in a number of ways. By having teammates frequently exchange information about their intended actions, they have the information necessary to consider each other's actions when choosing their own. Moreover, Hoplites encourages tight coordination by coupling the profit functions between teammates. Thus, a robot's decision making is directly influenced by its teammates' actions. Finally, the active coordination mechanism enables robots to explicitly negotiate tightly-coupled actions.

Two parameters, plan granularity and replanning frequency, determine 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. In doing so, it is better able to respond to slight changes in its teammates actions and thus coordinate more closely. 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 teammates actual activities more often. This allows it to react more quickly to unexpected changes in their actions. Both finer plan granularity and a higher replanning frequency result in tighter coordination.

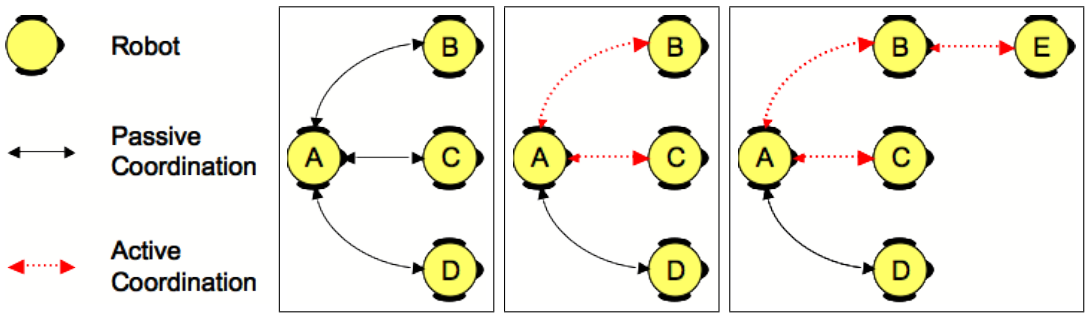


Figure 4.7: Illustrations of coordination topologies for a team of five robots. *(a)* A star topology in a passively coordinating group (A, B, C, and D). *(b)* A star topology in an actively coordinating group (A, B, and C). *(c)* A chain topology in an actively coordinating group (A, B, and E).

Planned Coordination

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. Hoplites produces planned coordination in both the passive and active coordination methods. In passive coordination, 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 active coordination, a robot intentionally coordinates its actions and its teammates actions and tries to purchase its teammates participation in a joint plan. Here, robots can *determine* the effects of their actions directly; this becomes necessary when the environment is complex.

Flexible Coordination

By "flexible coordination" we mean that a framework should not impose limitations on which robots can coordinate or how they can coordinate. Instead, it should allow the maximum flexibility so that the most effective solutions can be found. Hoplites is a flexible framework that facilitates coordination between multiple robots in many forms. Borrowing ideas from network and graph theory, we consider each coordination form in terms of its topology. Hoplites does not place restrictions on the topology of a team nor does it define the topology based on the needs of a single specific domain. Rather, it is general and can be applied to a number of problems. We illustrate a number of coordination topologies in the context of a team of five robots and depict some example scenarios in Figure 4.7. We use solid bi-directional arrows to indicate passive coordination and dotted bi-directional arrows to indicate intentional coordination.

We begin by considering the scenario in Figure 4.7 *(a)*, in which one single robot (A) coordinates simultaneously with multiple teammates (B, C, and D) and three robots (B, C, and D) each coordinate with a single teammate (A). Here, all robots are passively coordinating. 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 search terrain by making some portions of the space untraversable (hard constraints) or more costly to traverse (soft constraints).

In Figure 4.7 *(a)*, 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.

Next, we consider active coordination directed from one teammate (A) to several others (B and C) as depicted in Figure 4.7. That is, suppose A wishes to execute a particular plan by convincing both B and C to follow a team plan. 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 one of the planners we discuss later in Chapter 5). 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.

Lastly, we illustrate coordination in series between robots A, B, and E. Suppose that, unknown 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 4.7 *(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.

Other topologies are also possible. For example, we may have a tree topology (not shown) in which many robots are independently actively coordinating with a single robot which has accepted plans from all of them. These domain-independent examples illustrate several ways in which teammates can coordinate. Indeed, Hoplites does not make any restrictions on the number of teammates that can coordinate nor the type of coordination available to them. In principle, the most effective topology will evolve as teammates negotiate for the most profitable actions based on the requirements and constraints of their task. We demonstrate in Chapters 6 and 7 how a chain topology emerges in security sweep while a tree topology frequently emerges in LOSEX.

4.8 Meeting real-world requirements

To truly be useful, Hoplites must be applicable to many domains and be able to operate in the real world, where time for deliberation can be limited and where the environment can be changing and uncertain. In this section we illustrate how Hoplites meets the requirements for real-world operation.

Computational feasibility

Computational feasibility is the requirement that an algorithm or approach operate fast enough to respond to events and changes in a timely manner. In the simplest case, this means that robots must be able to plan at least as quickly as they move.

Hoplites achieves computational feasibility in a number of ways. First and most importantly, it distributes the planning of coordinated actions among the teammates whenever possible through the passive coordination mechanism. With this approach, teammates are at least able to do the best they can by responding to each other's actions. Secondly, this is often aided by distributed constraint structure that makes comparing different candidate plans very efficient. As we show in Chapter 6, passive coordination is fast enough to enable team members to consider a number of candidate plans using deterministic planners such as A*. Additionally, it is powerful enough to produce very good team solutions in many cases. Thirdly, Hoplites allows the team to be selective about when it expends computational resources towards improving team solutions. Active coordination is generally employed when the team solution is poor and when there is time for planning. It can be abandoned without system failure if it becomes too cumbersome.

Nevertheless, there are a number of design decisions that can greatly influence the computational requirements. First, the choice of planning algorithms can significantly affect the computation requirements to generate plans. 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. However, in our work, we find that optimal planners such as A* and D* [75] become computationally infeasible when search dimensionality is very large. We recommend and use search methods that sacrifice optimality for speed such as probabilistic roadmaps (PRMs) [76] or Rapidly Exploring Random Trees (RRTs) [77]. RRTs in particular have been shown to work well in high-dimensional spaces [78] and we use them in our own implementation to develop team plans in a computationally efficient manner. We discuss these algorithms further in Chapter 5. Secondly, selectiveness about when robots actively coordinate, with whom they coordinate, and how they coordinate can play a critical role in the computational requirements of Hoplites. We discussed these factors in Section 4.5.

Communication demands

Communication is critical for solving problems involving tight coordination because accurate information is a prerequisite to generating coupled solutions. For instance, in their work using two robots to dock a beam, Simmons et. al. [17] create a high-frequency communication link directly between robots' sensors to ensure timely information. Our goal for Hoplites is that its communication requirements should be on par with other systems that achieve tight coordination and also feasible for real robot teams.

Implementations of Hoplites will typically communicate three types of information to teammates: current state and environment, future plans, and requests for coordination. Because problems in our space are formulated with local constraint structures, this information is shared between a finite subset of robots. For instance, in a linked-arm structure, a robot only shares this information with at most two neighbors. Current state information is shared at a frequency that depends very much on the nature of the problem. For example, in a piano-moving problem, this frequency would be high because of the potential for disaster if imperfect information is used. Alternatively, in constrained exploration, this frequency can be lower because small changes in position should not affect the immediate outcome of the problem. Nevertheless, for a particular problem, Hoplites is as dependent on accurate information as any other approach would be and therefore should have the same communication requirements.

Unlike other approaches, however, Hoplites requires future plans to be shared between teammates. The communication requirements depend on the rate at which replanning occurs (which determines how many packets of information must be exchanged) and the planning horizon (which determines the size of the information packets). The replanning rate and the planning horizon, in turn, depend upon the nature of the problem and may have to be determined by trial and error. Despite these uncertainties, we believe Hoplites' computational requirements remain feasible for a number of reasons. Firstly and most importantly, the number of future plan messages scales linearly in the number of robots because of the local constraint structure. Secondly, the size of packets can be reduced in many cases by sharing low resolution versions of plans (i.e. rather than sharing path information at a resolution of 20cm, it is shared at a resolution of 1m) or by sharing low resolution versions for portions of the plan that are in the distant future. This is feasible because, in many domains, teammates need qualitative information about future plans (e.g. on which side of an obstacle a teammate is traveling) rather than precise quantitative information. We use these techniques in our implementation of constrained exploration on a real robotic platform. Thirdly, it makes sense to only share future plans when they change; otherwise, teammates can assume that the last plan is still in effect.

Finally, robots must communicate in order to actively coordinate. The three steps are re-

questing coordination, replying with price quotes, and purchasing or refusing participation. Price quotes and contract acknowledgments are inexpensive: they require only a constant packet size. Requests for coordination typically involve larger packet sizes since information about plans must be communicated. Still, this requirement is no greater than for regular plan announcements, though different information may be sent to each teammate. Moreover, because active coordination happens infrequently, this extra communication adds very little to the overall communication requirements

These properties, combined with our implementation experience, lead us to believe that the communication requirements are well within reasonable bounds for real robotic systems. We discuss them in detail for each of our implementations in Chapters 6 and 7.

Uncertainty

Uncertainty is inherent in every real-world domain. In tasks such as exploration and mapping, robots necessarily have incomplete information about their environments and the team's goal is to obtain this information. Additionally, information may be incorrect because of sensor error such as false positives from a laser scanner. Robots' estimated positions in the environment may also be imperfect because of actuation error. In multirobot systems, uncertainty in teammates' actions can also be an important component of overall uncertainty. The goal of all robotic systems is to operate in the presence of imperfect and uncertain information and to improve solutions when new, better information is obtained.

Market-based approaches including Hoplites are operational despite uncertainty because cost and utility functions can be evaluated and exchanges can be held even with only partial or poor information. As explained by Dias et. al. [37], this feature is one of the fundamental components of a market-based approach. In Hoplites, robots will use whatever information is available regarding the environment and the team at the moment to evaluate and select possible actions.

Nevertheless, poor information leads to suboptimal solutions in all approaches. In market-based approaches, in particular, uncertainty leads to imperfect estimates of the profit of certain actions and this causes robots to select suboptimal actions. For instance, in task allocation, poor information may lead to inaccurate bids for tasks, causing robots to perform tasks for which they are not best suited. Similarly, in Hoplites, imperfect information causes robots to incorrectly evaluate their candidate plans and to create team plans that may not be optimal or may be impossible to carry out given the true nature of the environment. Because robots' actions are tightly coupled in our problem space, poor decisions taken by one robot can have significant effects on its teammates and result in poor global solutions. In LOSEX, for instance, a robot may begin exploring an area believing that there are no obstacles near it that would cause line of sight obstructions; however,

once it arrives at the area, it discovers obstacles that will break connectivity. It is desirable, then, for robots to incorporate new information and update their plans to ensure that they always take the best actions given the information available and their computational abilities.

Hoplites has several mechanisms that enable robots to incorporate new information. First, a robot that is passively coordinating with its teammates can do this easily. Most simply, it can replan its own actions frequently. Also, by having robots share their intended actions frequently as well, their teammates have new information sooner and can incorporate it earlier. To efficiently use their computational resources to this end, robots should be selective and replan only when new information significantly affects the expected profitability of their actions.

Incorporating new information is particularly important when robots are actively coordinating since the interaction between teammates can be complex and critical to task success. As a general philosophy, we would like robots to be committed enough to fulfill the agreements made with teammates, while still having the flexibility to modify commitments when they are insufficient or even abandon commitments if new information reveals that they are actually detrimental to the mission.

We find that flexibility is maximized when robots are committed to the goals of the team plan rather than the specific plan itself. For example, suppose that in the constrained exploration problem in Figure 4.1 *(c)*, the bottom robot pays the top robot to take the dashed path around the obstacle as shown so that the bottom robot can reach its goal. Further suppose that the obstacle is actually much larger than it appeared to the robots when they made the agreement and that the top robot's current path will not allow the bottom robot to reach its goal. If the top robot is only committed to the explicit path agreed upon, its execution of this path will be pointless at best and costly at worst. If, instead, the top robot is committed to seeing the bottom robot to its goal and we allow it to modifying its plan freely provided it meets the needs of its teammate, then the robots can incorporate new information and efficiently improve their solution.

On the other hand, new information may reveal that the team plan is actually detrimental to the team's mission. This could occur if the cost of the final path the blue robot would have to take to help its teammate is more than the value of reaching that goal or more than the penalty that the robots would incur if they were disconnected from each other. If the relevant information had been available at the beginning, the agreement would never have been made in the first place. Now the robots must be able to abandon their plans. To achieve this, robots can continuously recompute and compare the cost and benefit of their coordination. If the costs exceed the benefits even after attempts to repair their solution (as described earlier), the group should disassemble and abandon its agreement. Once they are released from the contract, the robot that made the request may need to find a new

approach to achieving its goal or abandon its goal entirely (if that is an option), and its teammates are free to accept plans from other teammates or reconsider new plans from the original teammate.

Many exit mechanisms are possible for abandoning a plan: robots can agree upon compensation in advance in case of failed coordination or award compensation at the time of release and tailor it to the situation. Additionally, many compensation schemes are possible in which one robot bears all the cost, the cost is shared, and so on. In our experience, we find that an even simpler approach of teammates broadcasting their resignation from a plan when they perceive that the cost exceeds the benefits is sufficient and that compensation plans are not usually necessary.

As we demonstrate in Chapters 6 and 7, we have implemented these simple strategies in our experiments to allow robots to effectively operate in unknown environments and efficiently respond to new information about the environment and the team.

Generality

Lastly, to be truly useful, a coordination framework must be applicable to many problems and reusable from domain to domain. Hoplites meets the goal of generality in a number of ways.

Flexible Objective functions. Firstly, Hoplites inherits generality in part by being a market-based approach. Like most market-based approaches, one can apply Hoplites to a variety of different domains by tailoring the global utility function and the local profit function. This changes how the team evaluates their action choices and thus changes the problem they are trying to solve, without requiring a restructuring of the fundamental framework.

Flexible team structure. Secondly, as we discussed earlier in this chapter, Hoplites does not assume or depend upon any specific team structure. Hoplites does not restrict with which teammates a robot coordinates, with how many it coordinates, or how it coordinates. This is left to the user as a domain-specific decision. Additionally, Hoplites allows robots to dynamically change the team's structure to better respond to the mission's demands. Thus, Hoplites can be applied to tasks with very different team constraints.

Flexible planning algorithms. Thirdly, Hoplites neither is a planning algorithm nor requires any specific type of planning algorithm. Rather, it can accommodate any planner, from domain-specific algorithms that are tailored to the task to very generic planners that are reusable. In this way, Hoplites can be applied to any domain simply by giving it the

appropriate planning tools. The goal of Hoplites is to most efficiently utilize the planning tools provided.

Moreover, although Hoplites is especially designed for tightly-coupled problems that require planning, it can be easily applied to traditional multirobot problems. For instance, if we apply Hoplites to the LOSEX problem and then remove the tight coupling, we are left with the traditional exploration problem often solved by multirobot teams. (This is done simply by removing the penalty term from the global objective and the local profit functions.) Then, robots will always use passive coordination and will never use active coordination. The resulting solution will be the same as with other market-based approaches. Thus, Hoplites's ability to dynamically respond to the complexity of the problem enables it to solve *less-challenging* problems as well as more challenging ones.

4.9 Summary

In this chapter, we presented the general Hoplites framework. As we discussed in Chapter 1, planning for tightly-coupled multirobot systems is a difficult problem. Hoplites solves this problem efficiently by using distributed decision-making whenever possible and centralized planning as required. Hoplites is a market-based system that consists of passive coordination and active coordination mechanisms which are tailored to easier and harder problem scenarios, respectively. Passive coordination is light on computation and communication and allows teammates to iteratively respond to each other's actions without directly influencing them. When passive coordination traps robots in local minima, active coordination improves solutions by enabling robots to influence each other directly by buying each other's participation in complex plans over the market.

Hoplites successfully meets the demands of real-world domains and systems. First, because it selectively injects pockets of complex coordination into the system, Hoplites improves solutions while remaining computationally feasible. Second, Hoplites's communication requirements are on par with other approaches to tight coordination: robots must share the same state and environmental information in Hoplites as they would under any other approach. Hoplites further requires robots to share plan information and request active coordination, but the former scales linearly in the number of robots and the latter occurs infrequently. Third, the framework also operates in uncertain conditions and improves solutions with new information. Robots make initial decisions with whatever information is available, and they incorporate new information by replanning individual and joint plans and by exiting from prior commitments if new information reveals that they are detrimental to the team's mission. Lastly, Hoplites is general because the global and local objective functions, the team structure, and the planning toolbox can be tailored to any domain.

Chapter 5

A Planning Toolbox for Hoplites

In Chapter 4 we explained that Hoplites is a coordination mechanism and not a planning algorithm. That is, Hoplites decouples a robot's choice of planning algorithm from its choice of coordination mechanism, and it can utilize any planning algorithm or even a set of several planning algorithms as desired. This offers significant flexibility because it allows robots to choose planners according to the difficulty of the problem scenario rather than the complexity of the coordination mechanism. Moreover, when a simple planner fails to find a solution, robots can use more complex planners and thus attack problems with increasingly sophisticated algorithms.

Nevertheless, selecting appropriate planning algorithms is an important step to the team's success since these planners determine the quality of solutions. In this chapter, we discuss planning algorithms that, from our experience, are useful for solving problems in our space, and we explain how they can be incorporated into the Hoplites framework.

5.1 Planning with relaxed constraints

As we discussed in Chapter 4, different instances of domains from our problem space can be significantly easier or harder than others; this property is the basic motivation for an approach that selectively incorporates complex coordination. Moreover, although entirely trivial problem instances are rare, almost every problem instance has at least some subpart that is easy to solve. To maximize its overall effectiveness, we want Hoplites to efficiently solve these easy cases as well as the hard ones.

Detecting the difficulty of a problem scenario can be challenging since constraints are often complex and cannot be evaluated by inspection. In our experience, the best first step is to suppose it is easy, solve the problem with this assumption, and then check the effectiveness of the solution to evaluate whether the assumption was correct. If our assumption was correct, then we have solved the problem with minimal effort. If it was not correct,

then we still have a solution that we can use to guide our next steps. This kills two birds with one stone: it evaluates the complexity of the problem and simultaneously provides us with a potential solution.

This technique is called planning with relaxed constraints. Specifically, we first disregard the constraints between robots because, in the easiest cases (e.g. LOSEX in an obstacle-free environment), these constraints will be automatically met. Second, we optimally solve the remaining part of the problem (e.g. finding the shortest path to an unexplored area). Third, we check our solution against the full problem (e.g by checking the path for connectivity with other teammates' paths). Planning with relaxed constraints is useful because the planning can often be accomplished with very fast and simple algorithms.

As we discuss in Chapters 6 and 7, we use A* [74] frequently for this purpose because it is inexpensive, easy to implement, and provides guarantees on the quality of the solution. Most importantly, it often solves the larger problem of meeting constraints. Nevertheless, though such generic algorithms can be guided by domain-specific information (e.g. through the heuristic function), they cannot easily make use of complex, domain-specific intuition. We recommend additionally using domain-specific techniques that do exploit such information. Moreover, these can often be informed by the results of generic planners.

For example, in LOSEX, we know that if a particular path does not maintain connectivity with a teammate, then it is likely that no path in the same homotopic class will provide connectivity (for more, see Section 7.3). It is difficult to incorporate this intuition into A* but we have developed an alternative algorithm that is informed by information from previously-failed A* searches. Our algorithm compares a robot's A* path with its teammates' paths, finds points of line-of-sight failures caused by some obstacle, and generates a path to the goal via a waypoint on the far side of the obstacle. This guarantees a path in a new homotopic class, but does not guarantee a solution to the larger problem. Thus, this domain-specific approach also falls under the category of planning with relaxed constraints: it ignores the constraints with teammates, generates a solution, and then compares it afterwards. Domain specific planners such as this can often exploit simple algorithms and thus remain computationally competitive while enabling more informed searches.

5.2 Coupled planning

Planning with relaxed constraints clearly affords a number of advantages including speed, simplicity, and, in some cases, solution guarantees. In difficult problem scenarios, however, we cannot assume away the constraints between teammates and must incorporate the constraints directly into the planning process. This is particularly important during active coordination where we almost always have a complex problem and where robots have to purchase teammates' participation in coupled plans. In the most general case, this

means centrally planning the actions of a subset of teammates to ensure that constraints are met, which has complexity exponential in the number of robots and quickly becomes intractable. Yet, we have found two general techniques that make planning with constraints computationally feasible.

Reducing the search space

Problems in our space typically require planning through multiple dimensions, namely space and time. Unfortunately, search space grows exponentially in the number of dimensions; planning for two robots (x_1, y_1, x_2, y_2, t) involves a five dimensional space and can quickly become unmanageable as we increase our planning horizon. Fortunately, there are a number of techniques to reduce our search space that significantly increase tractability.

First, we can solve our problem using prioritized planning [79] in which we plan through dimensions of our search space sequentially rather than simultaneously. For example, we might solve our problem in the four dimensional space (x_1, y_1, x_2, y_2) first to get a set of plans and then synchronize these plans in the time dimension t. Specifically, in LOSEX, we might generate two paths, one for each robot, that belong in the same homotopic class and then plan a synchronized traversal of paths in time so the robots maintain line of sight contact.

Second, we can reduce the size of the planning space in each dimension and then plan through these reduced dimensions at once. For instance, we know in LOSEX that paths in the same homptopic class have roughly equivalent solution quality. Therefore, if we reduce the $x - y$ space to a roadmap of the different path homotopies, we should be able to find solutions in a fraction of the time. We have done this using Generalized Voronoi Diagrams [31] to produce a concise representation of the environment that allows us to plan efficiently through the five-dimensional space (x_1, y_1, x_2, y_2, t).

The unavoidable drawback to reducing our search space using either technique is that the optimal solution may be in the space we removed; so we are not guaranteed to find the best solution or indeed any solution at all. Nevertheless, without these techniques the problem may be intractable and we would have no solution anyway.

Randomized planning

We can also use sampling-based approaches to tractably plan in the full configuration space of the problem. By sampling the original search space instead of generating and planning over a discrete representation of this space, sampling-based planners efficiently provide solutions to problems involving very high dimensional configuration spaces. We have found that Rapidly Exploring Radom Trees (RRTs) [80] are especially useful because they are

easy to implement and to tailor to the domain; we use them in both of our implementations of Hoplites which we discuss in Chapters 6 and 7.

The drawback to sampling-based approaches, however, is that they are unable to provide guarantees of solution quality. So, they may find feasible solutions but cannot trade off different cost factors to find *good* solutions.

5.3 The Planning Toolbox

All of these planning approaches are useful for solving problems in our space, and Hoplites allows us to incorporate as many algorithms as we would like into a single implementation of the framework. The more algorithms we have in the planning toolbox, the more likely we are to find a solution. If we do not use these intelligently, however, they can stagnate the team by using excessive amounts of computation.

Experience has taught us that we can negotiate this tradeoff successfully if our toolbox consists of a small set of very different algorithms that are employed sequentially in order of increasing sophistication and power. A set of very different algorithms maximizes the chances of finding a solution by using all the tricks of the trade. A small set of algorithms forces us to eliminate similar approaches that would probably produce redundant solutions and thus waste computation. As a general guideline, we recommend incorporating *at least* one algorithm from each of the two general planning categories: planning with relaxed constraints and coupled planning. Simultaneously, we recommend using *at most* one planning algorithm from each of the specific planning categories: generic planners, domain-specific planners, reduced-space planners, and sampling-based planners. Finally, by using them in order of increasing sophistication, we use the most complex planners and the most computation only when it is truly needed, thus maximizing efficiency. This is consistent with the Quickest First principle developed by Borratt et al. [81].

5.4 Summary

Selecting appropriate planning algorithms is an important step in implementing Hoplites as these planners determine the quality of local solutions. In this chapter, we discussed a range of planning approaches that solve different scenarios in our problem space. These approaches fall into two categories: planning with relaxed constraints and coupled planning. Planning with relaxed constraints solves simple problems in which constraints are easily met. Moreover, this technique is efficient because very fast algorithms can be exploited that often provide guarantees on solution quality. Even when the resulting solutions fail to maintain constraints between teammates, they can be used to inform more complex algorithms. Coupled planning explicitly takes into consideration the constraints between teammates

during planning and thus can solve more difficult problems which require teammates to take complex joint actions. Moreover, we can make coupled planning efficient by reducing the search space and employing sampling-based techniques.

Hoplites allows us to incorporate many algorithms into a planning toolbox to maximize the opportunity to find solutions; the tradeoff is that excessive planning can stagnate the team. We recommend a small set of very different algorithms that are employed sequentially in order of increasing sophistication and power. Such a set spans the range of planning techniques but minimizes redundant computation by avoiding similar algorithms. By using the algorithms in order of increasing complexity, we use the most computation only when it is truly needed and thus maximizing efficiency.

Chapter 6

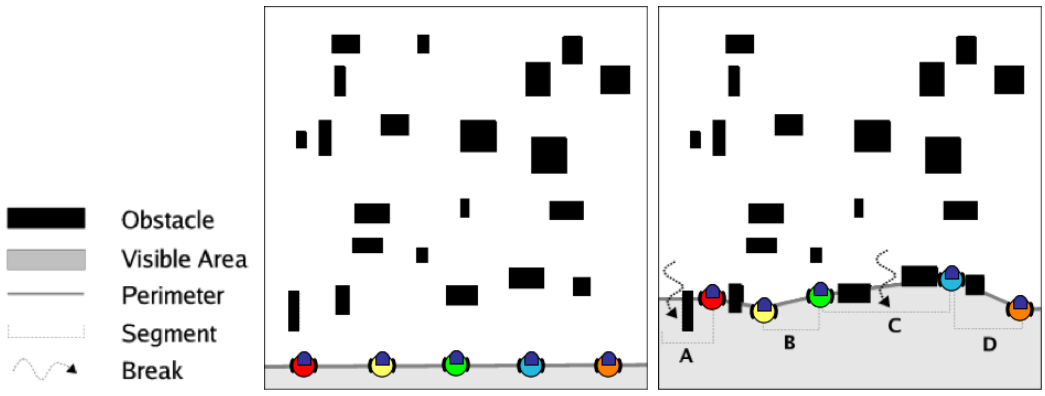
Experiments in Security Sweep

We have implemented Hoplites for the security sweep domain and the constrained exploration domain which we discuss in this chapter and the next, respectively. In combination, these different domains allow us to implement and explore the range of features we described in Chapter 4, including different objective and profit functions, team structures, planning methods, and coordination strategies. We use the security sweep domain in particular to demonstrate the impact of a linked-arm team structure, of chained active coordination for efficiently coordinating several robots, and of flexible contracts for operation in unknown environments.

We begin this chapter by describing the security sweep domain in detail (Section 6.1) and discuss related work in Section 6.2. We then illustratively explore how it falls into our problem space in Section 6.3. In Sections 6.4 and 6.5, we walk through our design process in detail. In Section 6.7, we present our comparison of Hoplites to three competing coordination frameworks and explore the results along several dimensions. Lastly, we highlight results on a team of indoor robotic vehicles in Section 6.8.

6.1 Domain description

In security sweep, a team of robots with a limited sensor range is tasked with sweeping a cluttered environment for mobile adversaries. The aim is to sweep the area as quickly as possible from one end of the environment to the other while preventing adversaries from remaining in the area undetected at the end of the sweep. Figure 6.7*(a)* is a sample environment containing twenty obstacles. The robots begin in a line at the bottom of the environment 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. The line of sight between the robots forms the *sweep front*: 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



(a) (b)

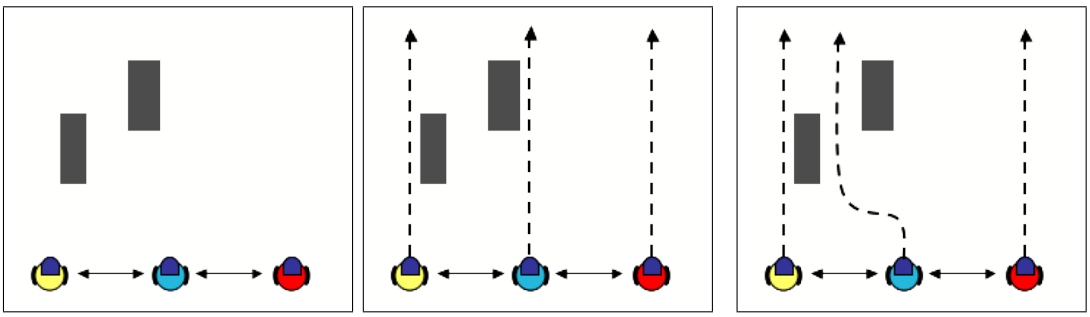
Figure 6.1: An illustration of the security sweep task. *(a)* A sample room to be swept. *(b)* Secure (B and D) and broken (A and C) perimeter segments.

minimizes the blind spots along this line that would allow a foe to enter undetected into the swept area. We refer to these blind spots as *breaks* in the front. Robots must coordinate to avoid these breaks as they traverse highly cluttered environments. Figure 6.7*(b)* illustrates the definition of a break. Segments A and C are broken because they intersect obstacles that would shield an adversary from view, enabling it to enter the secured area undetected. Segments B and D are not broken because all of the free space along the segments is observable by at least one robot.

Figure 6.2 shows how robots must coordinate to solve a security sweep problem. First, the easy but incorrect solution of robots following the shortest paths through the environment will create gaps through which an adversary could pass undetected (6.2 *(b)*). The problem can be solved if the center robot travels to the left to cover the center of the environment (6.2 *(c)*).

6.2 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, and 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. In our approach, the robots form a virtual line and when this line reaches the other end of the environment, all adversaries will have been discovered. Applications of sweep algorithms include the computation of Voronoi diagrams, map overlays, and nearest neighbors. Mehlhorn [82] provides a detailed description of the general algorithm and Van



(a) (b) (c)

Figure 6.2: An illustration of a security sweep problem with three robots and two obstacles in the environment *(a)*. An easy but incorrect solution emph(b) and the correct solution emph(c).

Kreveld [83] provides a thorough list of specific plane sweep applications and algorithms.

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 [84, 85]. However, nearly all of the work on pursuit evasion is aimed at developing purely geometric solutions 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 [30, 86, 87, 88, 89], permit only single searchers [90], or both [84, 85, 91, 92, 93].

One exception is work by Kim et al. [8] 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.

6.3 Coordination requirements

Through a series of illustrations, we demonstrate how security sweep fits in our problem space. First, security sweep requires multiple robots. Consider the simple environment in Figure 6.3 *(a)* where robot must sweep an area with one obstacle. Topologically, this environment is multiply connected, and it is impossible for one robot to guarantee that it has seen all adversaries because an adversary can hide indefinitely in the blind spot created by the obstacle.

Secondly, the domain requires tight coordination. In Figure 6.3 *(b)*, two robots must sweep a corridor with two obstacles. Although they only need to move in a straight line to

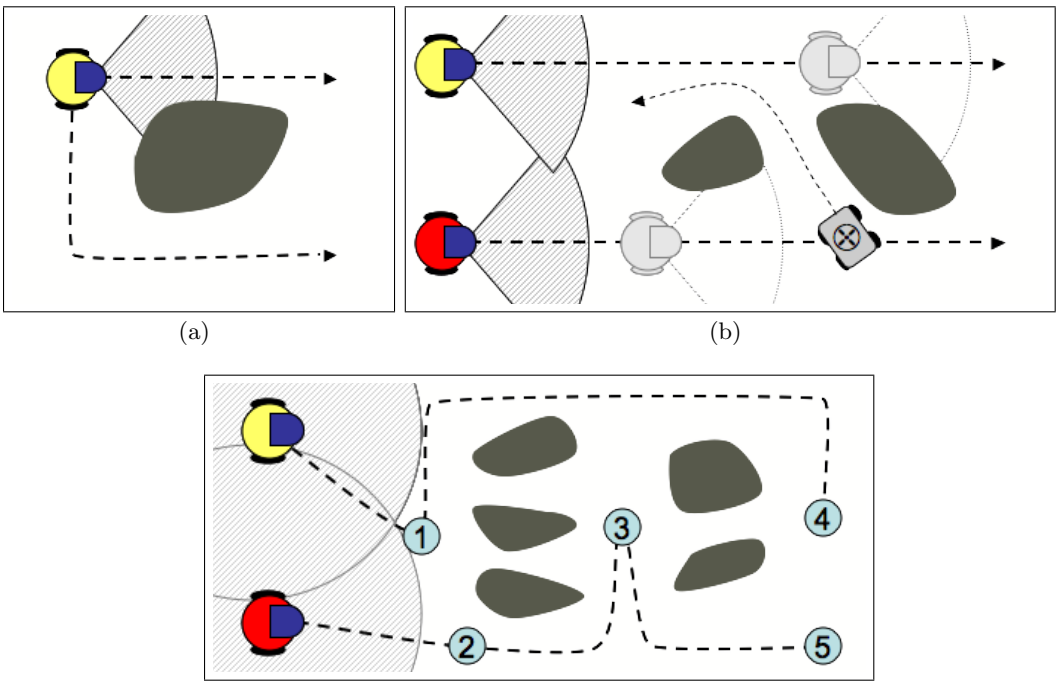


Figure 6.3: An illustration of the requirements of the security sweep domain. *(a)* A simple environment that cannot be swept by a single robot. *(b)* Although robots may be able to use a simple strategy to select optimal actions, they must tightly coordinate to time their trajectories properly. *(c)* 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.

clear the area, their timing must be right. If they remain in lock step, all adversaries will be detected. However, suppose the top robot moves faster than the bottom and at some time they arrive at the waypoints highlighted in the figure. Then, an adversary hiding in the environment could follow the dotted path and remain in the corridor without being seen by either robot. Thus, 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 6.3 *(c)*, the two robots must sweep a corridor with two groups of obstacles. A successful strategy is for the top robot to secure the already swept portions of the corridor while the bottom robot 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 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, the bottom robot cannot begin moving to waypoint 2 until its teammate has arrived at waypoint 1. When it arrives at waypoint 1, it must then wait for the bottom robot to move to waypoints 2 and then 3 before it can continue

to waypoint 4. Clearly, the two robots must plan their coordination in advance to complete the task; an emergent strategy would be insufficient in this environment.

If the robots 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, they have little or even no prior information. From their starting positions, they cannot see any of the obstacles and, for a significant part of the environment they can only see the first set of obstacles. Indeed, the perception and actuation for real robots is not perfect so robots may have uncertain or even incorrect map or state information. Therefore, as robots discover new information about the environment, they will have to drastically change their plans and reevaluate their coordination online. Robots must constantly replan their coordination to incorporate the latest information about their own states and the environment.

Finally, by reducing the range of the robots' sensors to be less than the width of the corridor, the two cannot alone clear the environment in Figure 6.3 *(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.

6.4 Design considerations: problem setup

To implement Hoplites on security sweep, we had to answer a number of questions raised earlier in the sections on design considerations (Sections 4.5 and 4.6). We discuss both the range of options for each decision point and the approach that we chose. These are summarized in Figures 6.4 and 6.5

Problem definition

Informally, a solution to the security sweep problem is good if it minimizes an adversary's opportunities to break the sweep front and enter the previously secured area. Given that we are using a plane-sweep method to clear the environment, it is easy to restructure the general problem as a set of locally manageable constraints. Recall that the sweep front is a chain of line segments linking every pair of adjacent robots and linking robots at the ends of the chain with the edges of the environment. Then, each robot is an endpoint of (at most) two segments in the sweep and so each robot's local constraint is to keep each of its segments intact. This formulation is both sufficient *and* necessary, which is unusual and which makes it a perfect mapping.

Problem Setup

1. **Problem definition:** Robots must traverse the environment while minimizing the total distance traveled and the mission time, and while keeping the sweep front intact. The sweep front can be described by a set of line segments between adjacent robots that must each be kept intact.

2. **Function C:** We use a cost function C instead of a utility function Q:

$$C = AT + B \sum_{r \in R} \sum_{t=0}^{T} Distance(r, t) + \Gamma \sum_{t=0}^{T} Breaks(t)$$

   where R is the set of robots, T is the mission duration, $Distance(r, t)$ is the distance traveled by robot r at time t, and $Breaks(t)$ is the number of gaps in the front at time t. Additionally, A, B, and Γ are $1, 1$, and 100, respectively.

3. **Function p:**

$$p(a, r, t) = \delta\,(y_t - y_{t-1}) - (\alpha\, t + \beta\, Distance(a) + \gamma\, t\, Breaks(r))$$

   where a is the action taken by robot r during a time step of duration t that moves it in the y direction from y_{t-1} to y_t, $Distance(r)$ is the distance traveled in that time, and $Breaks(r)$ is the number of breaks (maximum 2) in the robot's segments. $\delta = 40, a = 5, b = 5$, and $\gamma = 500$.

Figure 6.4: An outline of the problem setup design decisions made to implement Hoplites on security sweep.

The team's goal is to arrive at the other end of the environment. A good solution should also minimize the resources consumed by the team to perform the sweep. Although many resources might be of concern, we define resource consumption by the time taken and the distance traveled, which both reflect energy.

Quantification of user preferences.

We can measure each of our problem factors in a number of different ways. For instance, the connectivity of the sweep front can be measured by the number of segments which are broken, the amount of time during which the front is broken, and the size of the gap that would allow infiltration. We could also measure the time by the sum of individual times, the average time taken by team members, or the maximum time. The same options are available for distance.

We are interested in distinguishing a sweep front with one broken segment as better than a sweep front with five broken segments, and also a sweep front broken for one second as better than a sweep front broken for five seconds. Therefore, we measure the number of

Coordination Setup

1. **Passive coordination:** The candidate set contains seven paths from the robot's current location to seven goals that are each 50 steps from it in the $+y$ direction and evenly distributed between its neighboring teammates.
2. **Switching mechanism:** A robot switches to active coordination when all of these seven paths contain a break in the front.
3. **Active coordination:**
   a) Teammates: A robot attempts active coordination with the neighbor with which it shares the broken segment. In the case that the robot is an end robot, however, it coordinates with its only teammate.
   b) Team solutions: The robot uses an RRT to centrally plan for itself and its teammate to points that are beyond the obstacle causing the breaks in the front.
   c) Reserve path: No reserve path is computed.
   d) Market mechanism: Robots negotiate using directed peer-to-peer sales of plans.

Figure 6.5: An outline of the coordination design decisions made to implement Hoplites on security sweep.

break-seconds in the front during the course of the mission. Additionally, in many problems, robots must all operate until the mission is complete. Therefore, we measure time in terms of the overall mission completion time (the time at which the last robot arrives at the other end of the environment). Finally, the energy spent traveling is specific to an individual robot, so we sum the individual distances. This leads us to the utility function Q which formally quantifies our preferences:

$$Q = X - \left(AT + B \sum_{r \in R} \sum_{t=0}^{T} Distance(r,t) + \Gamma \sum_{t=0}^{T} Breaks(t) \right) \tag{6.1}$$

where X is the team's reward for completing the mission, R is the set of robots on the team, T is the time taken by the team to perform the sweep, $Distance(r,t)$ is the distance traveled by robot r at time t, and $Breaks(t)$ is the number of gaps in the front at time t. Additionally, A, B, and Γ represent the relative weights of the different factors in the equation. To make the mission challenging, we want the robots to expend both time and energy in order to avoid breaking the sweep front and therefore use a larger Γ relative to A and B. Specifically, we want the team to travel up to roughly 50 extra cells and to operate for 50 extra time steps to avoid one break the front for one time step. We can achieve this by setting A, B, and Γ to $1, 1$, and 100, respectively.

Notice that our reward X is a constant reward for the team regardless of the cost of

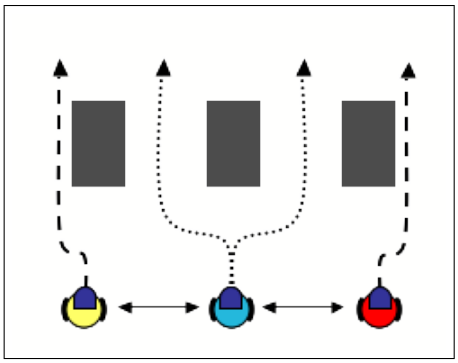


Figure 6.6: Assigning credit for actions is challenging even in security sweep. Here, three robots are tasked with sweeping an area, but this particular obstacle configuration makes it impossible for a team of this size with short sensor ranges. It is unclear which robot should be penalized for the break in the front.

their solution. Therefore, for simplicity, we can remove X from the equation and turn our goal of utility maximization into a simpler goal of cost minimization using the cost function C:

$$C = AT + B\sum_{r\in R}\sum_{t=0}^{T} Distance(r,t) + \Gamma\sum_{t=0}^{T} Breaks(t) \tag{6.2}$$

where all terms are the same as in the previous equation. Solutions will have the same preference ordering as in the utility maximization formulation, but we have removed terms that do not add to the evaluation. For simplicity, from here on we will evaluate solutions only using the equation C and not Q.

Mapping global utility to local utility

The next step is to create a local utility function which robots can use to guide their actions. Since the sweep front is a chain of line segments linking every pair of adjacent robots, each robot really only has control over the two segments of which it is an endpoint. Therefore we penalize a robot whenever either of its segments are broken.

Notice, however, that there are two robots in control of each segment but they may in reality play unequal roles in maintaining the front between them. For example, in Figure 6.6, the team is tasked with exploring an environment that is impossible for only three robots. In this scenario, the center robot has two options (highlighted as dotted lines), but both options will create penalties for it. Meanwhile, depending on what it choses, one of its teammates will incur a penalty while the other will not, even though they have essentially nothing to do with the outcome.

Although it is clear that robots play different roles and are responsible to different

degrees for breaks in the front, as this example demonstrates, it is unclear which robot is responsible at what time. Nevertheless, because the problem is symmetrical, on average the responsibility and the cost will be shared equally. We therefore believe that it is both reasonable and simple to penalize both robots equally, regardless of how the break was created. As we mentioned in Section 4.5, this gives the team a bi-directional linked-arm structure.

Now, we can return to the challenge of creating a local profit function for the robots. Capturing the distance component in C is simple: each robot incurs costs proportional to the distance it travels. This is consistent with the auction and bidding mechanisms described by Tovey et. al. [65] for the analogous problem of minimizing the sum of distances during task allocation. In general, it is more difficult to decompose the time component in C (which reflects the mission duration) since it is the maximum of all of the robots' times (known as the *makespan*) and not a simple combination of individual times. Nevertheless, notice that in this domain robots should take approximately the same amount of time to reach the other end of the environment because, first, they have roughly the same distance to cover and because, second, they must stay roughly in line to avoid breaking the sweep front. Therefore, the duration of the mission can be approximated by the average completion time of the teammates, which then becomes a function of the sum of the individual completion times. This allows us to use the same decomposition approach we used for distance: each robot is penalized in proportion to the amount of time it takes to complete the task.

This gives us a cost function c analogous to C that describes the profitability of a particular action:

$$c(a, r, t) = \alpha\, t + \beta\, Distance(a) + \gamma\, t\, Breaks(r) \tag{6.3}$$

where a is the action taken by robot r during a time step of duration t, $Distance(a)$ is the distance traveled in that time, and $Breaks(r)$ is the number of breaks (maximum 2) in the robot's segments.

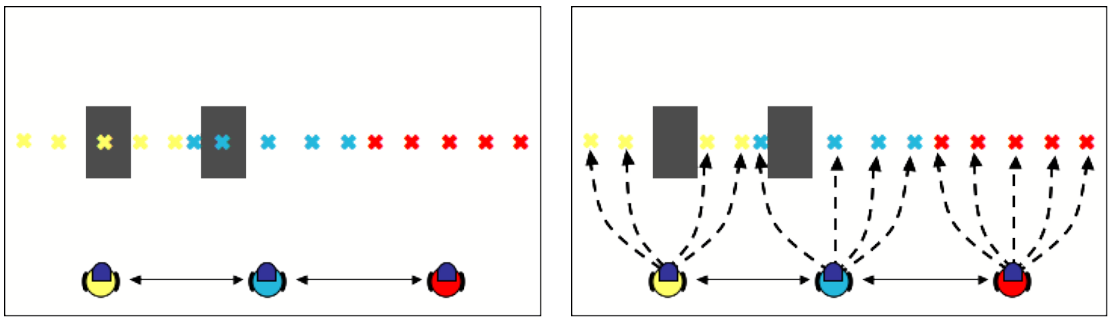
Nevertheless, one problem remains. Consider again the example in Figure 6.6 in which three robots cannot sweep the environment shown without breaking the sweep front. If we give the robots the option of staying in their current locations and not moving forward (a useful coordination feature), then it is conceivable that the most profitable action is to simply remain stationary to avoid the costs of breaking the perimeter. Indeed, the robots in this example could choose this option repeatedly and indefinitely push the imminent breaks beyond their planning horizon, and thus they might never complete the mission. To counter this possibility in a manner consistent with market-based approaches, we offer the robots rewards for moving forward so that their profit functions can incorporate the long-term goals of advancing across the environment. Specifically, we reward every robot

for each step it takes towards the end of the environment and penalize it for each step it takes backwards. Clearly this reward is not the same for every plan or for every robot, and therefore it has an impact on the solutions generated (unlike the team reward X that was in Q). This leaves us with the profit function p:

$$p(a, r, t) = \delta\,(y_t - y_{t-1}) - (\alpha\, t + \beta\, Distance(a) + \gamma\, t\, Breaks(r)) \tag{6.4}$$

where δ is the revenue generated by each advancing step and where all other terms are the same as in c. Since each robot is on average responsible for clearing $1/|R|$ of the width of the environment, we reward each robot $w \times 1/|R|$ of the environment for each step it takes, where w is the width of the environment. For example, in a 200×200 environment being swept by a team of five robots, each robot would receive \$40 for each step towards the other end of the environment. Clearly, this is somewhat arbitrary since only the relative values of α, β, γ, and δ matter, but it provides a reasonable starting point. To create consistency between the local profit function P and the global cost function C, the scaling terms α, β, and γ in P should have the same relative importance as A, B, and Γ in C. While we could set these terms to 1, 1, and 100 as in C, we must also consider the impact of δ which is absent in C. With this scheme in our example, each advance of 2.5 steps would offset one time step of a break in the front ($100/40 = 2.5$) and thus encourage robots to run through the environment rather than avoid perimeter breaks. The problem is that the weight of δ is too high relative to γ to retain our goal of making robots strongly prefer to avoid breaks. Therefore, we must scale δ down or γ up (and in turn scale α and β). In our experiments, we leave δ at 40 but scale α, β, and γ up by a factor of five so that they are 5, 5, and 500, respectively.

It may seem that many of these scaling decisions are somewhat arbitrary; in truth, they are. Remember that our goal in the experiments is to assess how different approaches perform given that they have analogous quality and cost functions and given that we offer them the same resources. (Different functions would unfairly advantage and disadvantage the approaches). Thus, for our purposes, it suffices to have functions that are internally consistent and that approximately reflect our goals. Nevertheless, in the real world, determining appropriate functions remain a challenging and important problem as it can have significant bearing on the success of any approach. While we can offer intuition and examples for selecting these functions, soundly and formally deriving them is an unsolved problem for all but a handful of relatively simple objective functions. This highlights that the credit assignment problem is itself primarily unsolved.



(a) (b)

Figure 6.7: To generate a set of candidate paths in this domain, robots first generate a set of candidate goals between their neighboring teammates *(a)*. Then, they discard any goal points that that intersect obstacles and then use A* to plan paths to the remaining goals *(b)*

6.5 Design considerations: active and passive coordination

Now that we have set up the problem and the team structure, we must deal with Hoplites-specific issues such as implementing active and passive coordination.

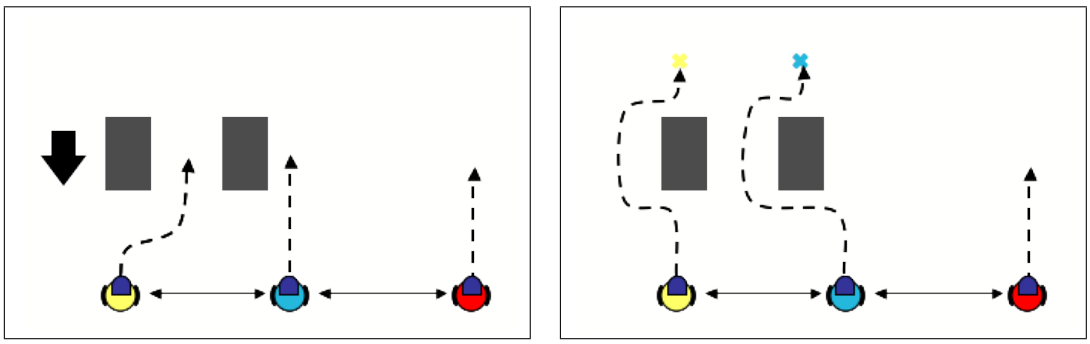
Passive coordination design

The first step of Hoplites is to implement passive coordination. In this domain, robots must only reach the far end of the environment, but they can arrive at any point along that edge. Since they have significant flexibility in how they move through the environment, we prefer a broader search of the space. Nevertheless, they ultimately want to advance through the environment so we can reduce computational waste by exploring only paths that create a net advancement rather than a net retreat. Additionally, it never makes sense for a robot to cross over its neighbors' paths, because their goal is most effectively met when they spread out and cover the entire sweep front. So, we can further reduce computation by only generating paths that do not overlap with teammates' paths.

In our implementation, we first choose a planning horizon h (explained below). Then, given that a robot r is at location (x_r, y_r), its left neighbor is at location (x_l, y_l) and its right neighbor is at location (x_r, y_r), we generate a set G of candidate goals, where the i^{th} goal is selected according to the following equation:

$$g_{xi} = (x_r - x_l)\frac{i}{|G|+1} + x_l \tag{6.5}$$

$$g_{yi} = y_r + hv \tag{6.6}$$



(a) (b)

Figure 6.8: An example of active coordination between two robots. The leftmost teammate can expect a break between itself and the environment's edge as shown by the black arrow *(a)*. By planning with an RRT to the goals beyond the obstacles, the robots can actively coordinate to solve the problem *(b)*.

where v is the velocity of the robot. Essentially we are choosing goal points that require approximately the planning horizon time to be reached and that are spread out equally between the robot's two neighbors. We discard any goals that lie in obstacles and use A* to plan to the remaining valid goals. This is illustrated in Figure 6.7.

Choosing coordination strategies

In security sweep, the method of generating candidate paths covers a wide range of possible solutions. Therefore, when a robot expects a break in one of its segments even after this search, we can be confident that it is because the problem cannot be solved without actively coordinating with its teammate(s). Therefore, robots attempt active coordination whenever they expect to experience a sustained break in the front.

Active coordination design

Choosing teammates

The first step of active coordination is to determine with which teammates to coordinate. The structure of this problem makes this an easy decision: a robot should actively coordinate with the neighbor with which it shares the broken segment since that is the only teammate that has direct control over the segment. Furthermore, in the case that an end robot (i.e. one next to the edge of the environment) expects a break, it has only one teammate with which to coordinate. This is the scenario illustrated in Figure 6.8 *(a)* where, given the middle teammate's path, the leftmost teammate can expect a break between itself and the environment's edge.

Planning team solutions

The next step is to determine how a team solution ought to be planned. For speed, we use a domain-specific implementation of the RRT algorithm discussed in Chapter 5. In our implementation, we select for each robot a goal that is directly ahead of it but beyond the obstacles causing the breaks in the sweep front (see the goals marked in Figure 6.8). We then use a centralized RRT to plan for both robots at the same time to synchronize their actions, while also taking into account the simultaneous positions of their *other* neighbors.

That is, suppose we have a scenario (not shown) with four robots r_0, r_1, r_2, and r_3 and further suppose that r_1 expects a break in its shared segment with r_2. Then, it will plan simultaneously for r_1 and r_2 while also ensuring that there are no breaks with the broadcast plans for r_0 and r_3. In our example, the left and center robots will generate a centralized plan that also ensures that there are no breaks between the left robot and the wall and the center robot and the right robot, without actually planning for the right robot. In a sense, it is a joint effort at passive coordination with r_0 and r_3. We must also decide which robot should produce the joint plan. In our domain, only information about at most four robots is required. Therefore, we believe either approach would work, and, for our implementation, have the robot that initiated the active coordination also develop the team plan.

The market mechanism

Lastly, we use very simple peer-to-peer sales of plans. A robot r_1 that wants its teammate r_2 to assist it will create a joint plan P_{12} consisting of its own path p_1 and path p_2 that it would like its teammate to participate in, and then proposes P_{12} to its teammate. r_2 quotes back to r_1 its change in profit if it were to accept this plan. This quote is essentially the difference between the local profit functions shown in Equation 6.4 evaluated over the length of the proposed plan. An absolute comparison of these values, however, would not offer an accurate picture of the difference in the plans because they may be of different lengths and may reach different places in the environment. Instead, we compute the average profit per step of the existing plan and extrapolate it to the length of the proposed plan. If r_1's profit margin allows it to compensate r_2, then it notifies r_2 who is then bound by contract to complete p_2.

6.6 Key experiment features

We use the security sweep domain to implement and evaluate several key features of Hoplites.

The linked-arm team structure

We use a linked-arm structure to model the team's global constraints. That is, each robot is always concerned with maintaining team constraints with exactly the same two teammates (except the "end" robots which have only one neighbor). There are several benefit to this approach. First, it is very simple: a robot always knows with which teammate to coordinate and it does not have to reason about the rest of the team. Secondly, it is scalable: the computation and communication requirements scale linearly with the number of robots. Third, it is consistent with and successfully models the line-sweep approach to the problem which we mentioned in Section 6.2.

Nevertheless, the one primary drawback is that it cannot be used to solve related problems such as pursuit evasion games because the line-sweep approaches themselves cannot solve them. For example, in an environment with concave obstacles or cluttered rooms, a solution might require robots to split off into groups to clear portions of the environment and then reconvene in a way that keeps the swept area secure. A fixed linked-arm structure would not be able to meet these needs.

In our experiments with constrained exploration, we use a flexible approach that overcomes these shortcomings and demonstrates Hoplites's ability to rearrange the local team constraints to meet the needs of the problem.

Bi-directional constraints

A related feature to the linked-arm structure is that we use bi-directional constraints between teammates. This means that a constraint between two robots is equally important to both of them and plays an equal role in their profit functions. In security sweep, a break in the front between two adjacent teammates results in equal penalties for both and thus both are concerned with keeping it secure. In contrast, in a uni-directional constraint system, one robot is more concerned with maintaining constraints than is its teammate. One way to implement this in a security sweep would be to only penalize the left end-point robot of a segment when it breaks.

Bi-directional constraints are useful because they increase the number of teammates concerned with maintaining constraints and thus increase the chances of finding solutions to hard problem scenarios. Moreover, they work well when there is a limited number of constraints to consider (in our case, two). These constraints, however, are not easily scalable in the general case. Consider a complex team structure in which constraints must be met between one robot and four of its teammates (i.e. in a star topology). With bi-directional constraints, that robot would have to consider and maintain all constraints simultaneously which may require excessive amounts of computation and communication. In contract, uni-directional constraints (in which only the four teammates try to maintain constraints

with the robot) might reduce both computation and communication without a significant loss in solution quality. We explore uni-directional constraints further for the constrained exploration domain in the next chapter.

Chained coordination

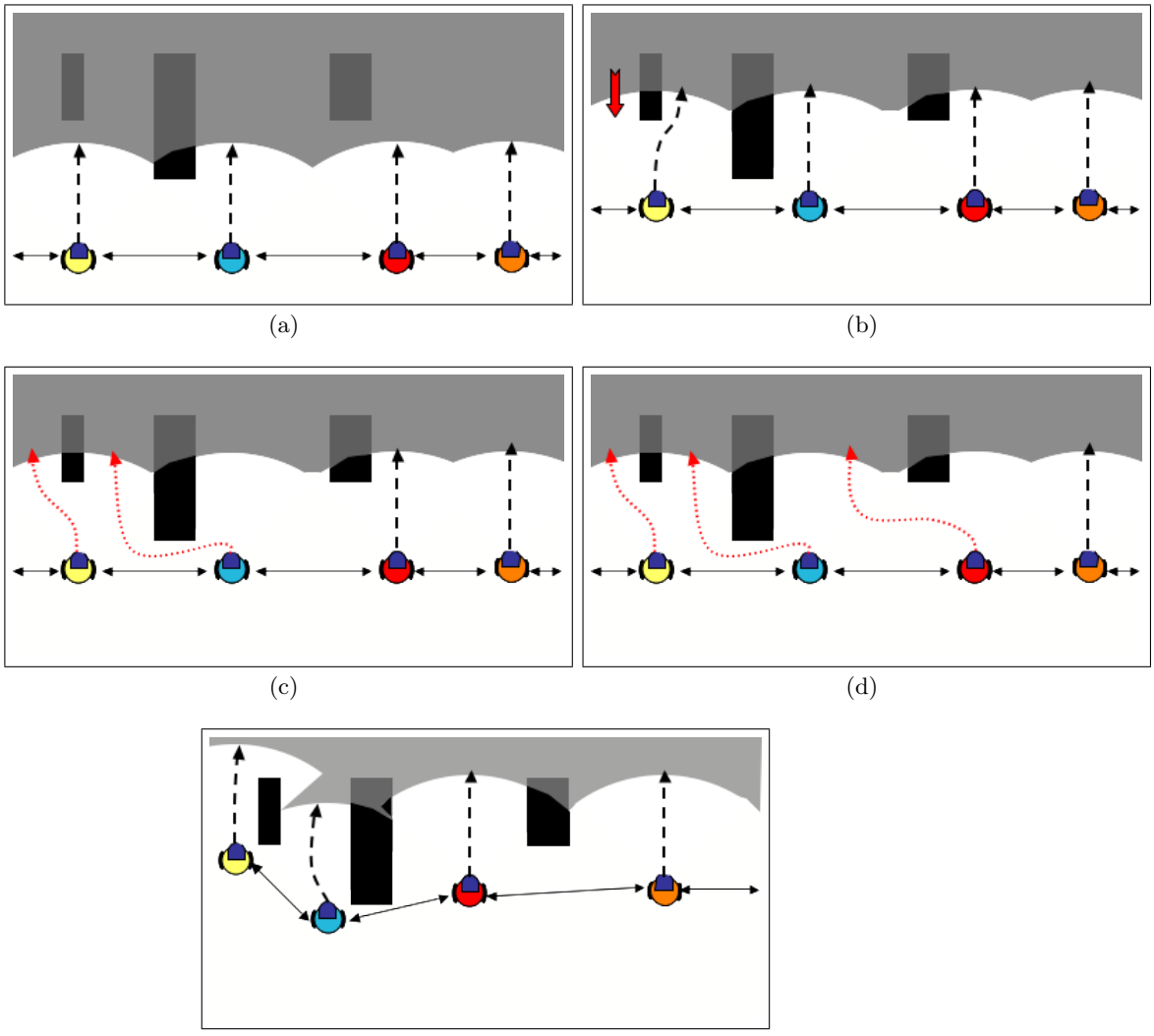


Figure 6.9: An illustration of the requirements of the security sweep domain. *(a)* A simple environment that cannot be swept by a single robot. *(b)* Although robots may be able to use a simple strategy to select optimal actions, they must tightly coordinate to time their trajectories properly. *(c-e)* 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.

In Section 4.4, we illustrated how chaining can be used to efficiently coordinate several

robots. We have implemented chaining for security sweep and demonstrate its impact in Section 6.7. Figure 6.9 illustrates how chaining is useful in the security sweep domain. Here, a team of four robots must sweep an unknown environment with three obstacles. The solid black lines between the robots highlight the perimeter front, the dashed black lines highlight their intended paths, and the dotted red lines indicate paths they are considering. Additionally, the solid rectangles are obstacles, area in white is known free space, and the area in grey is unknown space.

In this example, the four robots r_0, r_1, r_2, and r_3 begin sweeping using passive coordination (Figure 6.9*(a)*): each robot initially choses a straight path marked by dashed lines as its best path which keeps the sweep front intact. However, as the robots advance, they discover new portions of the environment. In Figure 6.9*(b)*, in particular, r_0 observes a new obstacle at the far left of the environment. The local environments of r_1, r_2, and r_3 have not changed, so they still initially choose the same straight path as before. r_0 must now choose between moving on the right or the left of the new obstacle. Given its teammates' simultaneous paths, both options result in the same penalty due to breaks, so the most profitable passive solution is to minimize cost, which results in the path to the right of the obstacle.

However, given that its best plan incurs a high penalty, r_0 tries to use active coordination to find a better solution. It finds that the team plan (illustrated with dotted red lines in Figure 6.9*(c)*) meets this end and proposes it to its teammate r_1. However, this plan creates penalties for for r_1 in the sweep front it shares with r_2, and compensating the penalty cost to r_1 is not affordable for r_0. To improve the solution, r_1 can also create a team plan for itself and r_2 as shown in Figure 6.9*(d)* which reduces r_1's cost of participation in r_0's team plan. Moreover, r_2's cost of participation is low since it doesn't require any front breaks. By aggregating its cost with r_2's cost, r_1 can quote a lower price to r_0. r_0 then pays r_1 and r_1, in turn, pays r_2; the team thus comes to the optimal solution shown in Figure 6.9*(e)*.

In this way, chaining enables a large set of robots to coordinate efficiently through the entire length of the sweep front and patch up potential breaks at one end of the sweep with participation from robots at the other end. In Section 6.7, we explore how often this happens in our experiments and its impact on the quality of the solution.

Flexible commitments

Finally, we explore the impact of flexible commitments in this domain. Because robots must sweep unknown environments, their commitments to each other must change in order to take into account new information. To deal with this, we allow a robot that has committed to a particular solution to change its actions whenever necessary, *provided it is not detrimental to the teammate with which it made commitments.* That is, a robot is free to take any

actions as long as those actions do not create breaks in the front between itself and its teammate. This strategy is flexible enough to allow robots to respond to new information and new demands from other teammates, but ensures that previous commitments are of paramount importance.

Nevertheless, it is possible that new information means that a robot is simply unable to keep the sweep front between itself and some teammate intact as agreed. In these circumstances, the robot is required to take actions that minimize the cost to its teammate even if this results in increased cost for itself.

6.7 Simulation experiments and results

We performed experiments in simulation comparing our implementation of Hoplites on the security sweep domain to three other frameworks. In this section we discuss in detail these frameworks, outline our experiments, and present and discuss our results. We also present our implementation of Hoplites on a team of Pioneer II-DX robots.

Frameworks

We used the MVERT framework [14] developed by Stroupe as a baseline for comparison. We further compared Hoplites to an improved version of MVERT which we call P-MVERT and to a system with passive coordination only.

The MVERT Framework. MVERT [14] is a behavior-based framework that was originally used for multirobot tasks such as foraging, tracking, exploring, and mapping. 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 intentional 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" [14]. 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.

We 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 sweep front, 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 front 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 move forward and are equally likely to take a northeast, north, or northwest step.

The P-MVERT Framework. 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 break will occur until it is one step away. By then, it is too late to avoid it. We improved MVERT to facilitate extended planning, resulting in what we have dubbed "P-MVERT" for Planning MVERT. In P-MVERT, a robot generates a set of candidate plans for itself and 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 MVERT is strictly a product of a longer planning horizon.

The Passive Coordination Framework. In P-MVERT, robots must guess their teammates plans. However, by allowing communication between robots, we can remove the guesswork. This brings us a framework consisting of just the passive coordination mechanism from Hoplites. That is, 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 breaks and to react to their teammates intended actions. We observed that passive coordination alone occasionally suffers from the instability: when two adjacent robots' planning cycles become synchronized, they may simultaneously change their plans and break the sweep front. They may also oscillate between two distinct plans in an effort to respond to each other's last broadcast plans.

Experimental setup

We 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 *(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. We also require robots to share information each time they move and each time they have a new plan. Plans, locations, and environmental information are shared only with immediate neighbors and the neighbors' neighbors (for planning during active coordination). Thus each piece of information is shared with at most four teammates. We prescribe that a robot moves at a rate of one unit per second (this roughly scales one map unit to one meter). Lastly, each robot replans its path every five seconds to take into account new information about the environment and its teammates, uses a plan lookahead of fifty time steps, and generates seven candidate plans during passive coordination using the method described in the passive coordination design (Section 6.5).

To make the domain challenging, we want robots to perform a sweep in which security is of the utmost importance. Specifically, we want robots to travel up to fifty units to avoid breaking the sweep front. Thus, our global cost function C is

$$C = 5T + 5 \sum_{r \in R} \sum_{t=0}^{T} Distance(r, t) + 500 \sum_{t=0}^{T} Breaks(t) \tag{6.7}$$

and so each step taken has a cost of at least 5 units (for the time required) and at most 10 units (for the time required and the distance traveled). Analogously, in our local profit function, we penalize a robot \$500 for each step that either breaks a previously-secure segment or fails to secure a previously-broken sweep front 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 (1/5th 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:

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

where $\text{Broken}(x_i, y_i, x_{i+1}, y_{i+1})$ is a function that returns 1 if the perimeter is broken during

	State	Env	Plans	Active Request	Active Reply	Active Accept
Msg Freq	RT	RT	$\frac{RT}{P}$	–	–	–
Actual Freq	10	10	2	–	–	–
Msg Size	$O(1)$	–	$O(L)$	$O(L)$	$O(1)$	$O(1)$
Actual Size	c	–	$50 \cdot c$	$50 \cdot c$	c	c

Table 6.1: Communication requirements of Hoplites on the security sweep domain. R is the number of robots on the team, T is the number of teammates with which each robot shares information, L is the planning horizon, P is the replanning period, and c represents some constant.

the transition and 0 otherwise. Note that time and distance costs are not equivalent if teammates wait for each other to "catch up" and therefore expend time but do not travel.

Communication Requirements

We can evaluate the communication requirements of Hoplites analytically from the problem setup. As listed in Table 6.1, we measure the frequency and size of different types of communication. First, if R is the number of robots and T is the number of teammates with which the robots share information, then, for our problem, there are RT messages about state information exchanged in each time step. Specifically, each of five robots move at a frequency of one step per second and each shares information with (at most) two of its teammates, resulting in 10 state messages sent per second. The size of state messages is constant: each robot shares an (x, y) position and its point in the current plan. Secondly, robots share information about the environment with each other so they have consistent pictures of the environment. Since new information is likely to be obtained at every step, the message frequency is the same as that of state information. Message size can vary greatly however; typically, robots only share new information obtained which depends on their sensor range and the nature of the environment. Thirdly, if robots communicate their plans each time they change and if P is the number of seconds between replanning attempts, then the number of replanning messages per time step is $\frac{RT}{P}$. Specifically, if five robots share new plans with two teammates every five seconds ($P = 5$) then two plan messages are sent in each time step. The size of the plan message depends on the planning horizon L; in our case, $L = 50$ and each robot shares a plan consisting of 50 (x, y) positions.

The communication frequency of active coordination cannot be determined before-hand; it depends upon the complexity of the problem. From our results in the next section, we find that there are only 2.34 instances of active coordination per experiment. For each instance of active coordination, one request is sent out that contains plan information required by the robot considering the proposal; this also depends on the planning horizon L. In general,

a coordination request contains 50 (x, y) positions and some constant information about the team plan such as when it will begin and end. The reply consists of a single numerical bid and the response to that bid is another constant-size accept or reject message.

Metrics

We measure the performance of the different approaches along a number of metrics. We measure the solution quality by the overall cost defined by the global function in Equation 6.7 and also evaluate the individual components in this equation. These individual components are the total number of break-seconds in the sweep front, the distance traveled by the team, and the time taken by the team to complete the mission. For each metric we evaluated the mean over the set of experiments and computed the standard error of the mean.

Note that the mission time includes only the execution time and not the planning time. We measure these factors separately because it is unclear how planning time (which is a result of the system hardware) would actually contribute to mission time and therefore to the overall solution cost on a real system. In contrast, factors such as the distance traveled and constraint violations should be accurately reflected since they are products of the algorithm and coordination framework [1].

Additionally, observe that there is a minimum cost incurred in order to complete every instance of a sweep mission which is the sum of the distance and time costs required for the team to travel across the environment as quickly as possible while ignoring front breaks. Just as we removed the reward factor from our global function Q because it was incurred by all approaches, we subtract this constant cost from each solution as well when we are comparing different solutions. This does not change our relative results but does remove constant terms that do not augment the comparison.

We measure the cost of generating solutions in terms of the planning time required by each robot. This includes the time spent planning individual and team paths, updating map information, communicating state information, and negotiating with teammates. We approximate the individual planning time from the total planning time (by dividing it by the number of robots on the team) because of compounding factors discussed earlier.

We also measured Hoplites-specific features related to contracts and active coordination. These include the number of instances of active coordination, the number of instances of chained coordination, and the length (in seconds) of active coordination between teammates.

[1] We are running experiments on a single computer; thus, it is unclear how planning time would change if each process was running on its own machine because of compounded factors such as paging. Moreover, the impact of planning time significantly depends on the relative speeds of the robot and the robot's CPU. A slow robot with a fast CPU would result in a small impact whereas a fast robot with a slow CPU would result in a large impact.

Finally, there are alternative metrics by which other coordination frameworks have been evaluated, such as optimality, convergence speed, and stability. Nevertheless, we believe that we have chosen metrics that are most relevant to our problem space. For example, we cannot produce the optimal solution to domains in our problem space because these problems are $\mathcal{NP}$-hard and intractable even for small teams; thus we cannot evaluate Hoplites on optimality. Convergence speed and stability are also not applicable metrics. Convergence speed measures how quickly a system can arrive at a final solution. While this is important for multirobot domains such as formation control [94] in which there is a desired goal configuration, it is not a suitable metric for domains such as security sweep where we are evaluating the entire course of the mission and where a desired configuration cannot be described analytically. Stability, in turn, measures how much inputs can be perturbed while still allowing the system to converge to a solution. Hoplites uses two types of inputs: the problem parameters and the system parameters. The problem parameters include the robots' initial states and the state of the environment and describe a particular problem instance; stability in this regard is not an appropriate metric because convergence is not an appropriate metric. System parameters, on the other hand, include weights on the different terms in our global and local profit functions. Stability in this regard *is* useful as it demonstrates Hoplites's sensitivity to imperfect implementation; therefore, we have explored this sensitivity experimentally and present the results in Section 7.6.

Discussion of Experimental Results

We are first interested in verifying a basic claim of this thesis: that the security sweep domain requires both planning and tight coordination. To examine the impact of planning, we review the improvement in performance between MVERT and P-MVERT; the only difference between these approaches is that MVERT has a planning horizon of one step while P-MVERT has a planning horizon of fifty steps. Figure 6.10 *(a)* compares the performance of the four approaches in terms of the average number of breaks in the sweep front. Error bars here and in remaining charts reflect the standard error of the mean. Notice that simply by increasing the planning horizon, we have reduced the number of breaks by about 50%. A greater lookahead allows the robots to make more informed decisions about which side of a particular obstacle to traverse, which, in turn, has a significant impact on the security of the front. Figure 6.10 *(b)* and Figure 6.11 *(a)* measure the average total distance traveled by the team in units and the total mission time in seconds. For P-MVERT, the statistically significant increases in distance traveled and in the mission time occur because robots may travel around the far side of an obstacle to keep the front in tact; but this increase is small because no single obstacle is very large relative to the length of the sweep.

To test overall whether the benefits of the increased lookahead outweigh the costs, we

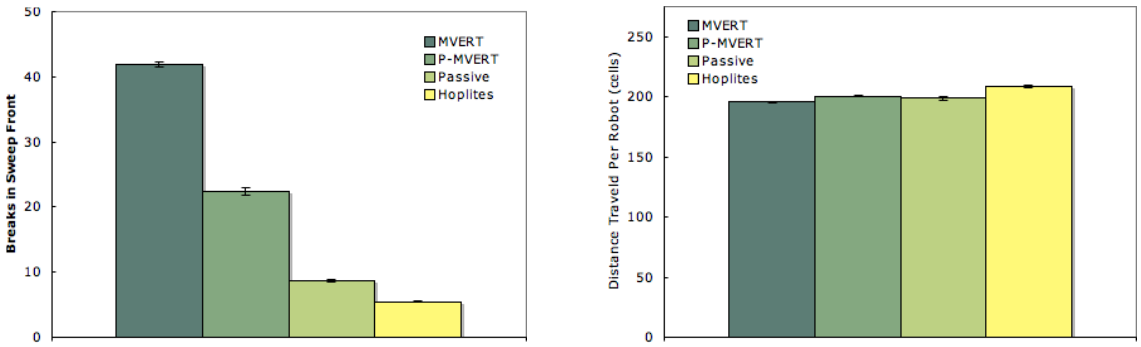


Figure 6.10: The performances of each approach in terms of the average number of violations *(a)* and the average distance traveled by a robot *(b)*. Error bars indicate the standard errors of the means.

must examine the impact on the global solution, which is a tradeoff among all the factors in which we have an interest. Using our global cost function in Equation 6.7, we can compute the cost of each approach over all the maps; these costs are presented in Figure 6.12. A comparison of these costs reveals that there is a large improvement in the solution cost between MVERT and P-MVERT; the cost has decreased by about 65%. This difference is due to the significantly higher importance we have placed on maintaining team constraints relative to the importance of completing the mission quickly. While the relative weighting may be somewhat arbitrary, the high importance of maintaining team constraints is a critical component of the complex problems we are trying to solve. Thus, we can conclude that planning is indeed a requirement for solving these domains well.

Finally, this improvement in solution cost from the increased planning horizon naturally results in an increase in planning time of about 17 seconds per robot as seen in Figure 6.11 *(b)*. While this is a significant increase relative to the planning time for MVERT, it has a minor impact in absolute terms: 30 seconds of planning time is acceptable for virtually any system and particularly for these problems.

We also want to confirm that these domains require tight coordination; we do this by comparing the performances of passive coordination and P-MVERT. Recall that the only difference between these approaches is that, in passive coordination, robots know which actions their teammates will take and do not need to estimate them. Our hypothesis is that if tight-coordination is required, then knowing teammates' actions precisely should improve robots' decision making and thus improve the team's solution. Conversely, if tight-coordination is not a fundamental factor in these domains, knowing its teammates' actions should not change a robot's actions and should not improve the team solution.

First notice that passive coordination reduces the constraint violations in the sweep front by over 60% as compared to P-MVERT. This improvement is even greater than the

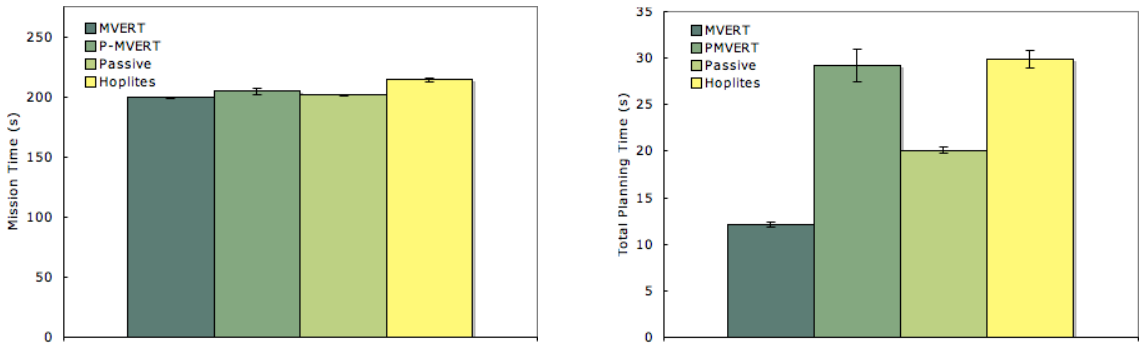


Figure 6.11: The performances of each approach in terms of the average mission completion time *(a)* and the average planning time per robot *(b)*. Error bars indicate the standard errors of the means.

improvement between MVERT and P-MVERT where we added planning. Second, this does not result in a significant increase in the distance travelled by the team or the time taken by the team because the set of paths from which the robots are choosing are the same; the robots are now just better at choosing which particular paths produce better solutions.

Third, the increase in communication results in an expected decrease in planning time (Figure 6.11 *(b)*) 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. Specifically, there is roughly a 9 second decrease in planning time for each robot.

As before, the final test is to examine the overall cost of solutions as shown in Figure 6.12). Here, we have another large solution cost decrease of over 60% compared to P-MVERT which confirms our hypothesis that the domains we wish to solve do require tight coordination between robots.

Note that in this domain we allow unlimited communication and do not consider it a resource we must conserve. Nevertheless, in some domains, we may want to limit our communication use, perhaps because bandwidth is limited or because we want our team to be stealthy. In these cases, communication should be factored into the cost function and we can expect to experience a less dramatic decrease in solution cost from P-MVERT to passive coordination since the cost decrease due to computation decrease may be offset by a cost increase due to greater communication use.

Finally, by comparing MVERT (the prior state-of-the-art) to passive coordination, we can evaluate one of the major contributions of this thesis: that passive coordination can produce tight coordination in a fundamentally competitive environment and solve many scenarios in our problem space. From MVERT to passive coordination, there is approximately an 80% decrease in the number of breaks in the sweep front, but effectively no difference in the distance traveled by the team; thus, robots are simply better at choosing

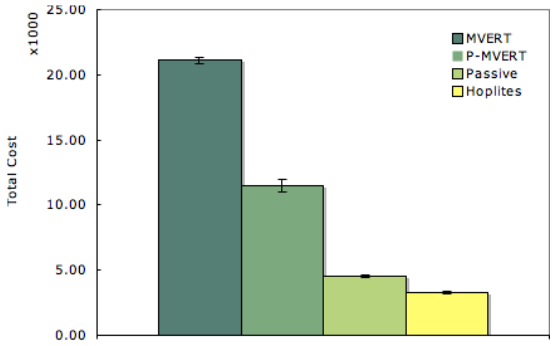


Figure 6.12: The performances of each approach in terms of the overall solution cost. Error bars indicate the standard errors of the means.

from the paths available to them. In sum, passive coordination reduces the overall cost of solutions by 80% as compared to MVERT. Moreover, passive coordination achieves this increase in solution quality with only a minor increase in planning time; just 10 seconds for each robot.

Finally, let us consider the performance difference between passive coordination and Hoplites, which is the most interesting comparison for two reasons. First, it substantiates our claim that passive coordination alone can trap robots in local minima. Secondly, it validates a major contribution of the Hoplites framework: that by actively influencing each other's actions by buying and selling participation in tightly coupled plans, the team can escape these local minima and arrive at significantly betters solutions. Hoplites creates an addition 60% decrease in violations in the sweep front. This comes at the expense of a small increase in distance traveled and in mission time since the team plans that prevent breaks in the sweep front are usually long and more complex than the plans generated by P-MVERT and during passive coordination. Overall, Hoplites decreases the solution cost by about 27% over passive coordination alone. The results demonstrate that, through the market, Hoplites provides a powerful mechanism that allows robots to actively work together to find better solutions. Moreover, this benefit requires an increase in computation time of only 8.5 seconds because robots only plan through high-dimensional joint action spaces (*i.e.* actively coordinate) when it is expected to improve the team's solution.

Finally, we observe some interesting trends if we examine these results in more detail. Figure 6.13 *(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 our base approach MVERT; indeed, we expect nothing less. More interestingly, we observe that the effect of enabling active coordination differs between environments: in some environments, this difference is very small (e.g. environment 7) and in others it is very large (e.g. environment 19).

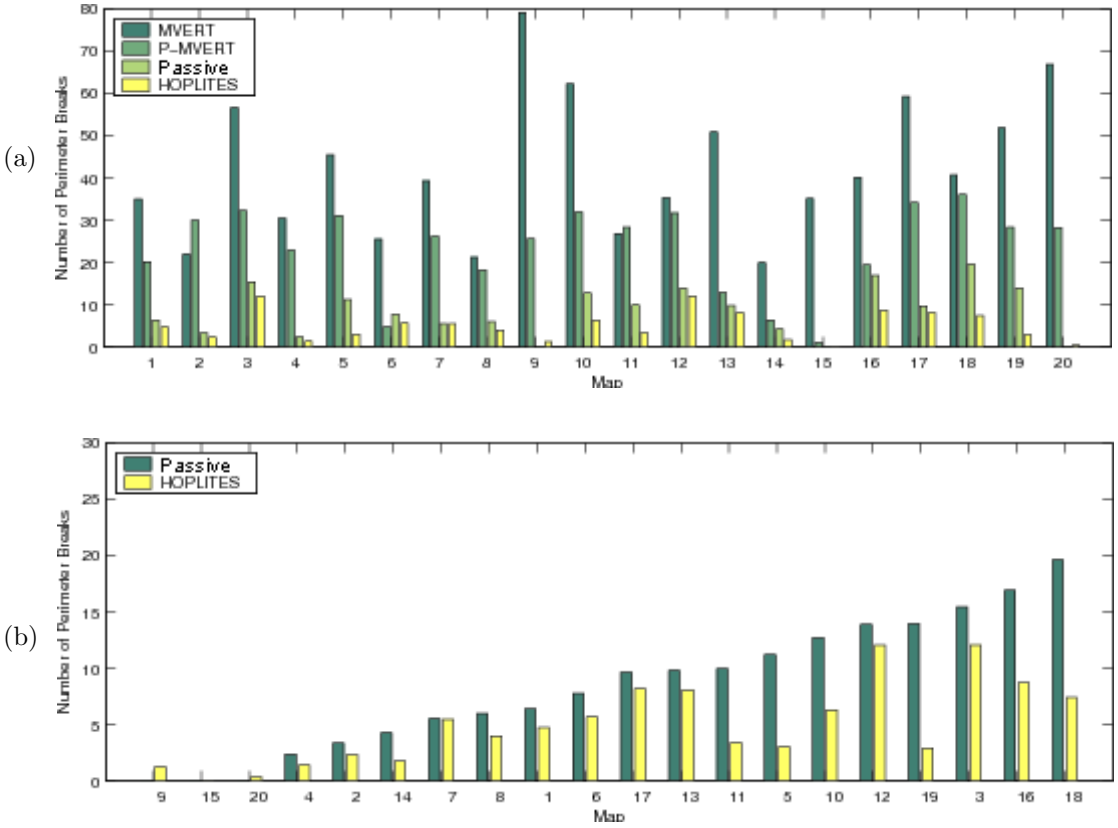


Figure 6.13: The performances on individual environments. *(a)* The number of perimeter breaks caused by MVERT, P-MVERT, passive coordination, and Hoplites on individual environments ordered by number. *(b)* The number of perimeter breaks caused by passive coordination and Hoplites on individual environments, ordered by increasing complexity. Error bars have been left out for clarity.

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 since it depends on the complex and non-obvious configuration of obstacles. Nevertheless, we can use the performance of passive coordination as an approximate measure of complexity: the more complex an environment, the poorer the performance of passive coordination. Figure 6.13 *(b)* compares the performance of passive coordination and Hoplites where environments are ordered by decreasing performance of passive coordination which correlates with increasing environmental complexity.

First, note that Hoplites outperforms passive coordination alone 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 E and a "hard" set H based on passive coordination's performance. Then the average number of instances of active coordination per experiment on E and H are about 1.56 and 3.15, respectively. Active coordination occurs over twice as often on H than on E. Moreover, 7.1% of these instances are part of a chained effort in H, whereas only 2.7% are in E; this is over a 2.5-fold increase from E to H. In sum, robots are working significantly harder on harder environments than they are on easy environments. The payoff for this is clear: the number of front breaks incurred by Hoplites relative to passive coordination is 0.76 on set E and 0.54 on set H. In terms of overall cost, the average solution cost ratio of passive coordination to Hoplites on E is 0.95 and on H is 0.64.

This difference in how and when active coordination improves the overall solution exists in some part because there is simply more room for improvement with active coordination when we are dealing with difficult scenarios. 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 passive coordination alone slightly outperforms Hoplites, the average number of perimeter breaks for both approaches is less than two, suggesting very easy environments.

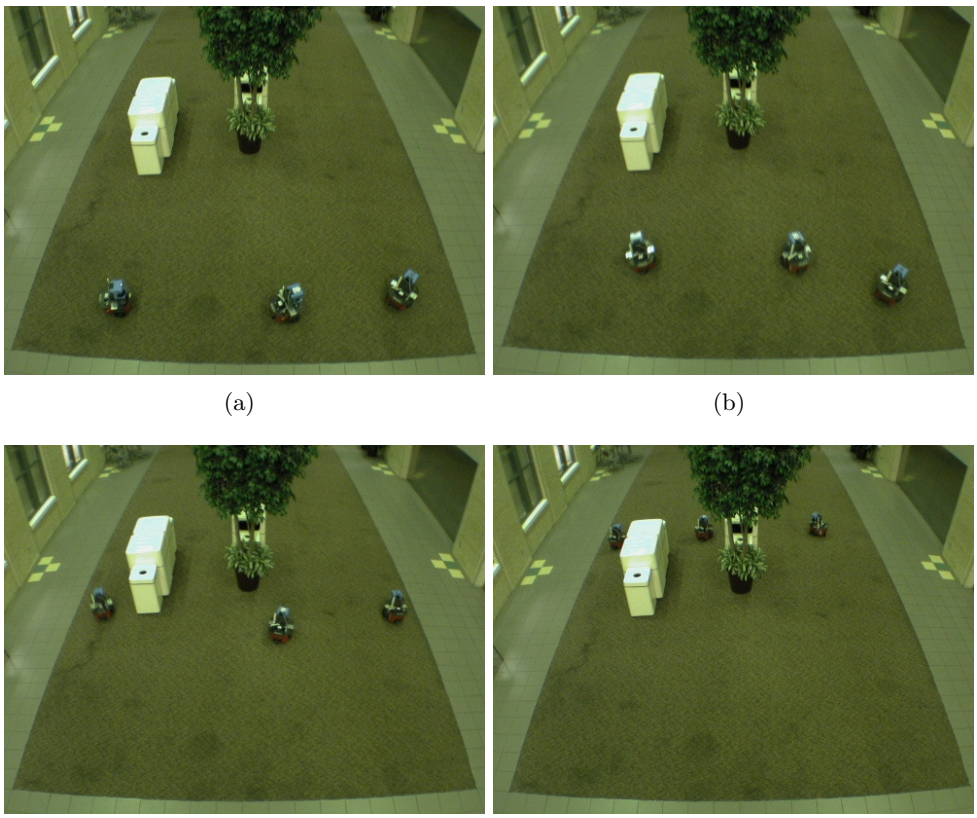
6.8 Validation on indoor vehicles

Figure 6.14: Three Pioneer II-DX robots used for our experiments.

We implemented Hoplites on a team of Pioneer II-DX robots shown in Figure 6.15. The on-board CPU is a 266 MHz Mobile Pentium R processor and has 256MB of memory. Each robot is further equipped with a SICK LMS 200 laser range finder that provides 180^o fields of view, KVH E-Core 1000 fiber-optic gyroscopes for localization, and an 802.11b wireless

ethernet card. Details of the supporting low-level controls and sensing can be found in the TraderBots implementation by Dias et. al. [95].

The robots' task was to sweep the 7 × 10 meter environment pictured in Figure 6.15 *(a)* with two obstacles. This environment was consistent with the example environment shown in Figure 6.8 *(a)*. As we described, the left-most robot actively coordinated with the center robot to complete the sweep without breaking the line of sight. Active coordination began between the left and center robots in subfigure *(c)*, where the center robot moved to the left of the center obstacle. The team completed this task in under two minutes.



(a) (b)

(c) (d)

Figure 6.15: Snapshots (in chronological order from left to right) of a team of three Pioneer II-DX robots completing a security sweep task in the Newell-Simon Hall atrium at Carnegie Mellon.

6.9 Summary

In this chapter, we described our implementation of Hoplites on the security sweep domain and outlined in detail a number of design considerations. This domain highlighted a few

features in particular: the use of a linked-arm team structure, bi-directional constraints, chained coordination, and flexible commitments.

Our experiments demonstrated a number of features about both the domain and the Hoplites framework. Incorporating communication and planning into MVERT [14] transforms the behavior-based approach into passive coordination and results in an 80% reduction in solution cost. This demonstrates that the security sweep domain does require both planning and tight coordination to be solved effectively. It also supports our claim that passive coordination can produce tight coordination in a fundamentally competitive environment and solve many scenarios in our problem space. The full Hoplites framework further reduces solution costs by 27% compared to passive coordination alone. This demonstrates that robots can successfully escape local minima and improve global solutions by by buying each others participation in complex plans over the market. Because Hoplites selectively injects pockets of complexity, it provides these improvements while remaining computationally competitive with other distributed approaches; specifically Hoplites required only 8.5 more seconds of computation time. Lastly, by forcing the robots to discover the environment and operate without a prior map, we have shown that Hoplites is successful even in uncertain conditions. We also validated Hoplites on a team of small mobile robots performing a security sweep task.

Chapter 7

Experiments in Constrained Exploration

We have implemented Hoplites on the constrained exploration domain as well as on the security sweep domain. In constrained exploration, a team of robots must explore an environment by visiting a set of target locations while meeting some communication constraints. This domain maps to a number of other real-world domains, and we have used it to develop a more general version of Hoplites that uses a flexible team structure and a powerful planning toolbox to maximize responsiveness to task complexity. Additionally, we formally evaluate Hoplites's sensitivity to imperfect information and to parameters chosen during implementation, and we evaluate its ability to scale to large teams and its performance relative to new competing approaches.

We begin this chapter in Section 7.1 with a formalization of constrained exploration in general and of LOSEX in particular, and we simultaneously review the relevant literature . We then explore how LOSEX falls into our problem space in Section 7.2. In Sections 7.3 and 7.4, we walk through our design decision process in detail. In Section 7.6, we present our comparison of Hoplites to two competing coordination frameworks and explore the results along several dimensions. We also present data evaluating key features of Hoplites such as the flexible team structure and the planning toolbox. Lastly, we highlight results on a team of large outdoor robotic vehicles in Section 7.7.

7.1 Domain description and related work

The general constrained exploration domain can represent a number of different problems. In this section we first formalize the domain as a Multi-Depot Traveling Salesman Problem (MD-TSP) and then express our LOSEX problem in MD-TSP terms. We simultaneously discuss related work to explain our problem formulation.

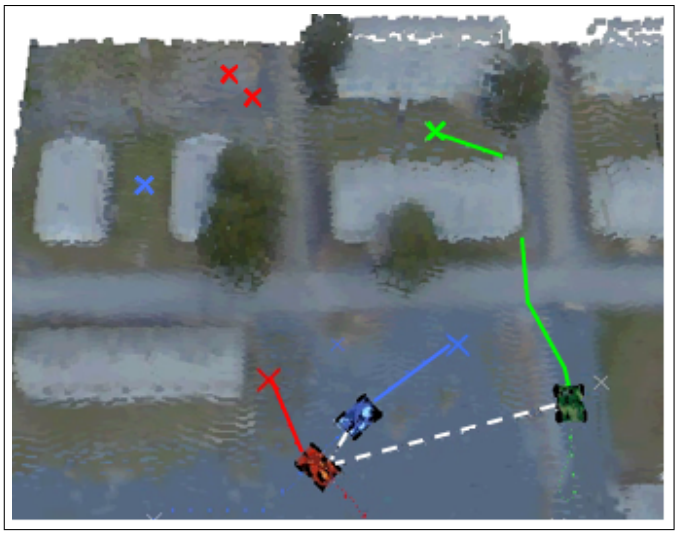


Figure 7.1: An example of constrained exploration from our simulator with three robots exploring an outdoor environment with trees and buildings. Each robot has been assigned a set of cities indicated by the matching colors.

Formalization of constrained exploration

We formalize general constrained exploration as a special case of the Multi-Depot Traveling Salesman Problem (MD-TSP) [96, 97] in continuous space. The MD-TSP is defined by a set S of salesmen that start at different depots, a set of cities C, and some cost for traveling between cities. Each city must be visited by one salesman, and the total distance traveled by the salesmen must be minimized. To solve the MD-TSP, the cities must be allocated to the salesmen and each salesman must construct a tour of minimum cost. MD-TSP has been shown to be $\mathcal{NP}$-hard [37].

For constrained exploration, the salesmen S correspond to members of the robot team R and the cities correspond to exploration sites (we use the term "sites" and "cities" interchangeably). We then describe the communication cost at any point during execution by some function *commcost* and require that this cost cannot exceed some threshold $\mathcal{T}_c$. By varying *commcost* and $\mathcal{T}_c$, we can create a variety of constrained exploration problems and also map constrained exploration to a number of other domains such as communication-constrained reconnaissance, visibility-constrained pursuit evasion and security sweeping, and distance-constrained transportation.

In formulating constrained exploration as an instance of MD-TSP, we have a problem that closely couples task allocation, decomposition, and execution. As such, it is an interesting and complex domain that allows us to explore a wide range of coordination issues within a common problem formulation [39]. To focus our efforts on meeting tight-coordination and planning requirements, we omit the task decomposition and allocation components of the

general problem. Specifically, we pre-allocate a set of sites to each team member and the challenge is for each robot to visit its set of sites while maintaining communication constraints with its teammates. It is as if we have a team of heterogeneous robots and each site can only be visited by one particular robot on our team (perhaps because the robot is uniquely equipped to perform a task at that site). This is illustrated in Figure 7.1 where robot's colors are matched with the sites they have been assigned. We will revisit the complete problem of coordinated decomposition, allocation, and execution in Chapter 8

LOSEX and related work

We can formalize in constrained MD-TSP terms the LOSEX problem in which the team must maintain a connected network determined by line-of-sight contact. Our cost function *commcost* evaluates a robot's ability to communicate with every team member:

$$commcost(r_i) = \sum_{r_k \in R} nolink(r_i, r_k) \tag{7.1}$$

where $nolink(r_i, r_k)$ returns 1 if there is no communication link between robots r_i and r_k and 0 if there is. By setting $\mathcal{T}_c$ to 0, our formulation requires the team to be fully connected. Notice that evaluating the function *nolink* requires accurate information about the team's entire communication network; moreover, estimating it in the future requires accurate information about every robot's future position and about the entire environment. This is consistent with our informal description of the problem in Chapters 1 and 4. It is clear that by defining the problem in this way, we can only effectively solve it for small teams using centralized approaches that have total information and can plan for all team members simultaneously.

This assessment is consistent with the prior work on this problem. Ferguson et. al. [98, 99] demonstrate that sampling techniques such as Rapidly-exploring Random Trees (RRTs) [80] can centrally solve this problem for teams of up to ten robots. Kalra et. al. [31] show how centralized planners such as A* can be coupled with roadmaps to enable the efficient search of a reduced planning space. Wagner and Arkin [6] use a hybrid approach in which a central controller with complete team and environment information selects a pre-determined plan for exploring an area according to the immediate task features and transmits to each robot a recommendation for action. Although robots take actions independently, the controller monitors them closely and dictates plan progress. Vazquez and Malcolm [100] also use a distributed approach to a closely related problem in which communication attenuates with range but not through obstacles. Each robot maintains a complete map of the team's network and attempts to explore unknown areas provided it wont disconnect the team. Given that it is infeasible in a distributed system for each robot to also maintain a map of the

team's future network (which is necessary for long-term planning), robots use a very short lookahead to choose their next actions.

None of these approaches scale to large teams as they all require at least one agent to have complete team and environment information at every time step. Because Vazquez and Malcolm's approach is distributed [100], it requires *every* teammate to have this complete information. Given that this approach cannot scale and also uses short-term planning, it is likely that centralized approaches (and even distributed approaches with long-term planning such as Hoplites) will easily outperform it. The centralized approaches [98, 99, 31, 6] all suffer from single points of failure. Additionally they can become intractable for large teams because the size of the planning space grows exponentially in the number of robots. Wagner and Arkin's enumerated approach is additionally not adaptable to arbitrary and complex instances of the problem.

Given that LOSEX is not feasible or tractable in the general case, many researchers restructure the problem into a set of locally manageable constraints. To our knowledge, all such formulations use a fixed-link structure like the one we described in Chapters 1 and 4. In this structure, each robot r_i must only be able to communicate with its teammate r_{i-1}; if this is true for all i, then the team is connected. We can formulate this as a case of MD-TSP where *commcost* evaluates a robot's ability to communicate with its designated neighbor:

$$commcost(r_i) = \text{nolink}(r_i, r_{i-1}) \tag{7.2}$$

where $\text{nolink}(r_i, r_{i-1})$ returns 1 if there is no link between robots r_i and r_{i-1} and 0 if there is. Again, by setting $\mathcal{T}_c$ to 0, our formulation requires the team to maintain a connected chain. This formulation is distributable, tractable both in terms of communication and computation, and much more easily evaluated.

Nguyen et al. [101] use a distributed behavior-based system to solve this problem for a single teleoperated robot with a team of dedicated relay nodes that increase its communication range and thereby increase its operating range. Robots use teammate-following behaviors to tightly coordinate and maintain line-of-sight with their designated neighbor. Schouwenaars et al. [41] formulate the problem as a mixed integer linear program (MILP) to provide both tight coordination and planning. This approach suffers from a single point of failure, has complexity exponential in the number of robots, and may be unresponsive to changes in the environment. However, it does take advantage of the simple evaluation of the solution. Esposito and Dunbar [40] use the same structure to solve LOSEX. In particular, they mathematically prove that this formulation requires all robots to traverse paths in the same homotopc class [102]; essentially, every team member must pass on the same side of every obstacle in the environment. In doing so, they highlight an important limitation of

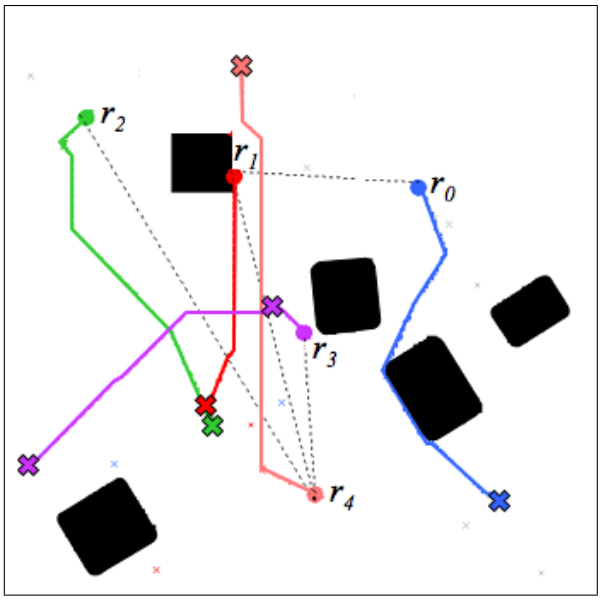


Figure 7.2: A snapshot from our simulator in which five robots operating in an unknown environment have replaced their original linked-arm tree structure with a complex tree structure. Black areas are obstacle, solid lines indicate the robots paths to their respective targets (marked as x's), and dashed lines indicate the communication connectivity between the robots.

this structure which we touched upon in Chapter 4: that it overly constraints the team and limits the solutions they can reach.

We use a problem formulation that is significantly more flexible even though it is only slightly more complex than the linked-arm formulation. We use almost the same *commcost* function as defined in Equation 7.3, but we relax the requirements for r_i's neighbor:

$$commcost(r_i) = \text{nolink}(r_i, r_k) \tag{7.3}$$

where $r_k = r_{i-1}$ at the beginning of the mission, but r_k can change during the course of the mission provided that doing so does not disconnect the team. Assume that every robot r_i keeps track of just one possible communication route c_i by which it can pass a message to r_0. At the start of the mission, $c_i = \{r_{i-1}, r_{i-2}, ..., r_0\}$ for all $i > 0$. Then, if r_i wants to replace its current neighbor with teammate r_k, it can do so if $r_i \notin c_k$. If r_i is in c_k, then swapping could cause a circular network that disconnects the team. (Note that switching to r_k in this case would not *necessarily* cause a disconnect: because we are only keeping track of one route, r_k may have an alternative but untracked route that does not use r_i). On the other hand, if r_i is not in c_k, then we can guarantee that r_k has an alternative route to r_0 and thus swapping to r_k cannot disconnect the team.

Unlike checking for team-wide connectivity, several features make checking this property

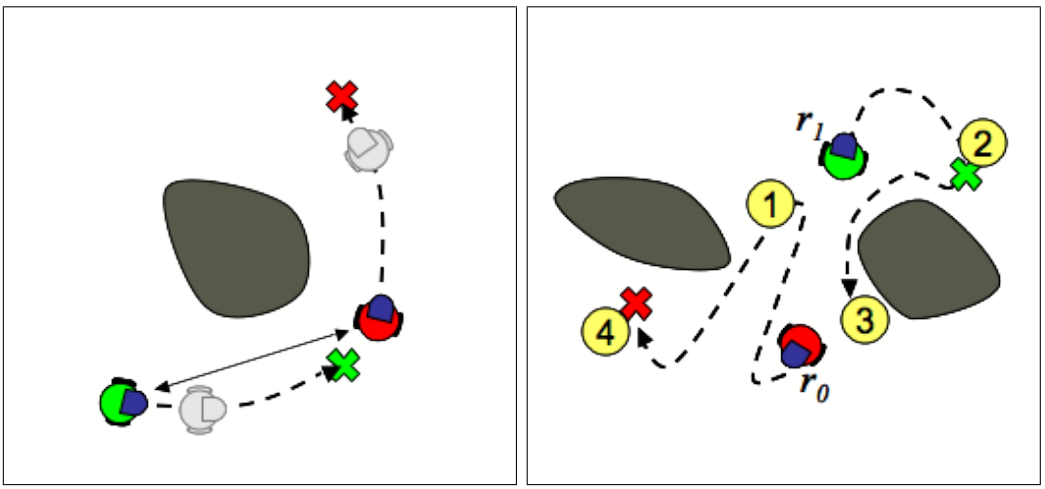
tractable. First each robot only tracks one route to one teammate. This requires at worst $O(n)$ communication and computation, if the team is in a linked-arm structure. Even then, only one robot (the end robot) would have a route using all of its teammates. Moreover, in complex environments, the team is likely to evolve into a tree-like structure which is $O(log(n))$. Secondly, robots only need to swap neighbors (and therefore perform this operation) when they lose contact with their current neighbor. Thirdly, r_i can only consider switching to those teammates that it can directly see, which further limits the number of route messages and computations. Finally, a robot's route is only utilized by its teammates during swapping; depending on the rate of swap inquiries, we can further save by having robots compute their routes only when they are required via message propagation.

With this formulation of flexible local constraints, we allow robots maximum flexibility. For example, in Figure 7.2, robots have replaced their initial fixed-arm structure with a complicated tree structure that allows them to visit their target sites in a very cluttered environment. This solution would not be possible with a fixed arm.

Finally, let us mention other prior work that solves constrained exploration in simplified ways; while we do not believe these approaches can successfully solve complex instances of this problem, they may optimize along other parameters. Powers and Balch [103] have created VBCP, an MVERT-style approach to keeping communication contact. This approach requires prior map information to estimate connectivity and uses very short-term lookahead. While myopic decision making prevents it from providing good solutions, VBCP is very fast and responsive.

7.2 Coordination requirements

We previously used constrained exploration to illustrate many concepts related to Hoplites in Chapter 4. We briefly discuss how LOSEX, in particular, falls into our problem space. First, even simple scenarios require tight coordination between teammates. In Figure 7.3*(a)*, the two robots each have a target site which they can reach easily while maintaining line of sight contact; nevertheless, if they do not remain in lock step, they could lose line-of-sight communication around the obstacle as shown and violate team constraints. Second, in complex scenarios, the domain requires planned tight coordination. In Figure 7.3*(b)*, each robot must traverse a complex path to ensure that connectivity is maintained while its teammate visits its target region. Here, the robots must visit the marked waypoints in order: once r_0 arrives at waypoint 1, r_1 can visit its target at waypoint 2; then, r_1 must arrive at waypoint 3 so that r_0 can visit its target at waypoint 4. This solution requires planned coordination between the teammates to discover the solution and then tight coordination to execute it effectively. These features make LOSEX a representative candidate from our problem space.



(a) (b)

Figure 7.3: Solving LOSEX requires tight coordination *(a)* and planning *(b)*. The robots' targets are marked by x's, their paths are dashed lines, and their waypoints are circled numbers. Dark grey shapes are obstacles in the environment and light grey outlines of robots represent their possible future positions.

7.3 Design considerations: problem setup

There are a number of design considerations related to implementing Hoplites on LOSEX. In this section we discuss the problem setup and in the next section we discuss Hoplites-specific design considerations. These are summarized in Figures 7.4 and 7.5

Problem setup

Through our formalization of constrained exploration and our discussion of LOSEX implementations in Section 7.1, we have already done most of the work in setting up our problem. Note that an optimal solution to our LOSEX problem consists of a set of tours that meet three requirements. Firstly the tours ensure that each robot visits each of its target sites once. Secondly, the tours are of minimum cost. Thirdly, the tours are coordinated in a way that ensures that each teammate maintains contact with its current neighbor or selects a new neighbor in a way that ensures team connectivity. Except in the simplest cases of completely clear environments, however, tours of minimum cost do not simultaneously meet communication constraints. In reality, finding tours of minimal cost and maintaining communication constraints are competing requirements: robots may have to deviate significantly from the minimum cost tours by taking longer paths to provide connectivity. Thus a tradeoff is necessary between path cost and connectivity with neighbors.

If a robot has a good solution to the traveling salesman problem, it can appear to have

Problem Setup

1. **Problem definition:** Robots must complete pre-computed city tours while minimizing the total distance traveled and the mission time, and while maintaining line-of-sight constraints with their current neighbors.

2. **Function C:** We use a cost function C instead of a utility function Q:

$$C = A\,T + B \sum_{r \in R} Distance(r) + \Gamma \sum_{t=0}^{T} \sum_{r \in R} Disconn(t, r, N(r))$$

   Here, R is the set of robots, T is the mission length, $Distance(r)$ is the distance traveled by r, and $Disconn(t, r, N(r))$ determines whether r is connected to its neighbor $N(r)$ at time t. Additionally, $A = 1, B = 1$, and $\Gamma = 100$.

3. **Function c:** We use a local cost function c instead of a profit function p:

$$c(a, r_i, t) = \begin{cases} \alpha\, t + \beta\, Distance(a) & \text{if } i = 0 \\ \alpha\, t + \beta\, Distance(a) + \gamma\, Disconn(r_i, N(r_i)) & \text{otherwise} \end{cases}$$

   where robot r_i takes action a of time duration t. $Distance(r_i)$ is the distance traveled in that time, and $Disconn(r_i, n(r_i))$ indicates whether r_i has contact with its uplink $N(r_i)$. Additionally, $\alpha = 1, \beta = 1$, and $\gamma = 100$.

Figure 7.4: An outline of the problem setup design decisions made to implement Hoplites on constrained exploration.

good performance even with poor coordination because it's disconectivity will be offset by a low cost tour. Alternatively, a poor TSP solution can disguise good coordination because of its high cost. Finally, robots can also avoid coordinating by finding alternative solutions to TSP and re-ordering their tours. To concentrate on the problem of coordination and to lessen the impact of solving TSP, we give our robots a pre-computed optimal tour of the cities and require that they visit the sites in the order they appear in the tour. We can think of robots having to perform a series of strictly ordered tasks at each site. Thus, each robot starts with the minimum cost tour and must determine how to get from city to city in a way that balances the need to arrive as quickly as possible while also maintaining connectivity along the route.

Finally, we must define what we mean by the "cost" of the tour. For consistency with the multirobot vehicle routing literature [104, 105], we define the cost of a tour as the distance that the tour covers. To ensure connectivity to neighbors, robots may also need to wait for their neighbors at particular locations while they visit target sites. Thus, in contrast to unconstrained MD-TSP, minimum distance tours in LOSEX are not necessarily minimum time tours because robots may remain stationary at different points in the tour.

Coordination Setup

1. **Passive coordination:** The candidate set contains only the shortest path from a robot to its next city in the tour.
2. **Switching mechanism:** A robot switches to active coordination when the shortest path contains constraint violations with the current uplink and no new uplink can be found.
3. **Active coordination:**
   a) Teammates: A robot attempts active coordination with the current uplink.
   b) Team solutions: The uplink tries to plan a solution that provides connectivity to the robot's existing path. The uplink uses a rage of planning algorithms from A* to RRTs.
   c) Reserve path: The robot simultaneously plans its best alternative path using the same planning toolbox.
   d) Market mechanism: Robots negotiate using directed peer-to-peer sales of plans.

Figure 7.5: An outline of the coordination design decisions made to implement Hoplites on constrained exploration.

While this doesn't increase the distance traveled, it may increase mission completion time which, in turn, affects energy consumption for the whole team. Therefore, we also track the duration of the mission.

Quantification of user preferences.

The next step in our problem setup is to define a tradeoff between our factors of interest. To this end, we must first define how to measure the team's inability to maintain constraints. For instance, we might use a binary valuation in which the team is either connected or disconnected. For our implementation, we measure the degree of disconnectivity at a particular time by the number of disjoint subsets of teammates at that time. This is a more expressive measure as it gives us a sense of *how* disconnected the team is. We measure the cost of tours in a manner consistent with typical MD-TSP: the team's tour cost is the sum of the costs of individual tours. Additionally, the team's goal is to visit sites, so we sum the number of sites visited by each robot. Finally, we define mission duration as the time at which the last robot completes its tour. This gives us our global utility function Q:

$$Q = X \sum_{r \in R} Sites(r) - \left(AT + B \sum_{r \in R} Distance(r) + \Gamma \sum_{t=0}^{T} \sum_{r \in R} Disconn(t, r, N(r)) \right) \quad (7.4)$$

where R is the set of robots on the team, T is the mission length, and $N(r)$ is robot r's neighbor. Additionally $Sites(r)$ is the number of sites visited by robot r, $Distance(r)$ is the distance traveled by r, and $Disconn(t, r, N(r))$ determines whether r is connected to $N(r)$ at time t. Finally, X, A, B, and Γ are the relative weights of the different terms in the equation, and we use them to encode our preference for the tradeoff between minimizing tour cost and minimizing disconnections in the team. As in the security sweep problem, we want to make the mission challenging so we set A, B, and Γ to 1, 1, and 100, respectively. Thus, a robot should travel up to about 50 extra cells to avoid violating constraints with its neighbor. Later in Section 7.6, we offer insight into the impact of these weights through a set of sensitivity experiments. Lastly, note that every solution requires robots to visit each site, so the term $X \sum_{r \in R} Sites(r)$ is a constant; we can remove it from the equation for simplicity and encode our preferences as a global cost function:

$$C = A\,T + B \sum_{r \in R} Distance(r) + \Gamma \sum_{t=0}^{T} \sum_{r \in R} Disconn(t, r, N(r)) \tag{7.5}$$

All terms are the same as in Q, only the reward term has been removed. Through the remainder of this section we will refer to the quality of a solution in terms of cost.

Mapping global utility to local utility

We must now map the global cost function C in Equation 7.5 to local utility functions that define the robots' individual preferences for different actions. The first challenge is to solve the credit assignment problem. Clearly, each robot is responsible for its individual contribution to its tour duration and tour distance. Each robot also has control over maintaining communication links with its neighbor and should be penalized whenever that contact is lost. It is unclear, however, whether or not this constraint is symmetric. That is, if a communication constraint is being maintained between two robots r_0 and r_1 and that link breaks, should r_0 or r_1 or both be penalized? In security sweep, each robot had equal control over both of its sweep front segments, and, if we had formulated LOSEX conservatively using a linked arm structure, the same would be true in this case. By allowing exchanges of neighbors, however, this problem becomes asymmetric. As highlighted by Figure 7.2, robots can form complex tree-like structures in which one robot may be a communication node for many teammates (e.g. r_5) while another robot may not be a node for any teammate (e.g. r_3). Moreover, a robot does not have control over which teammates select it as their new neighbor or leave it for a different neighbor, or when they do so. Therefore, constraints cannot be symmetric: each robot must only ensure that it can communicate with its neighbor; it does not have to ensure that other teammates can communicate with it.

Given that constraints are uni-directional, we can assign simple and clear terms to the problem that reflect this asymmetry: if a robot r has chosen to maintain contact with teammate t to satisfy its communication constraints, we refer to t as r's *uplink* and to r as t's *downlink*. Additionally, recall that any robot can select a new uplink provided it is not a node in the potential uplink's route to r_0. This means that there will always be at least one other robot that has r_0 as its uplink. Because constraints are assymetric, r_0 does not need to maintain any uplink for the team to be connected. We call r_0 the "root" of the team. In summary, if every robot ensures that it can communicate with r_0, then the team is connected and r_0 does not have to ensure connectivity with any teammate. This leads us to two different local cost functions:

$$c(a, r_i, t) = \begin{cases} \alpha\, t + \beta\, Distance(a) & \text{if } i = 0 \\ \alpha\, t + \beta\, Distance(a) + \gamma\, Disconn(r_i, n(r_i)) & \text{otherwise} \end{cases} \tag{7.6}$$

where a is the action taken by robot r_i during a time step of duration t, $Distance(r_i)$ is the distance traveled in that time, and $Disconn(r_i, n(r_i))$ is a boolean indicating whether r_i has contact with its uplink $n(r_i)$. Here, α, β, and γ are analogous to A, B, and Γ in Equation 7.5. That is, α is 1, β is 1, and γ is 100. We discuss these weights again in Section 7.6

Lastly, by allowing robots to have different cost functions, we have seamlessly created a heterogeneous team of robots where individual robots have different preferences and are targeted for different activities. This is a strength of market-based approaches in general. Certainly our team is not nearly as heterogeneous as a team with physically different robots that have different capabilities, but it highlights that Hoplites can be implemented on a diverse team.

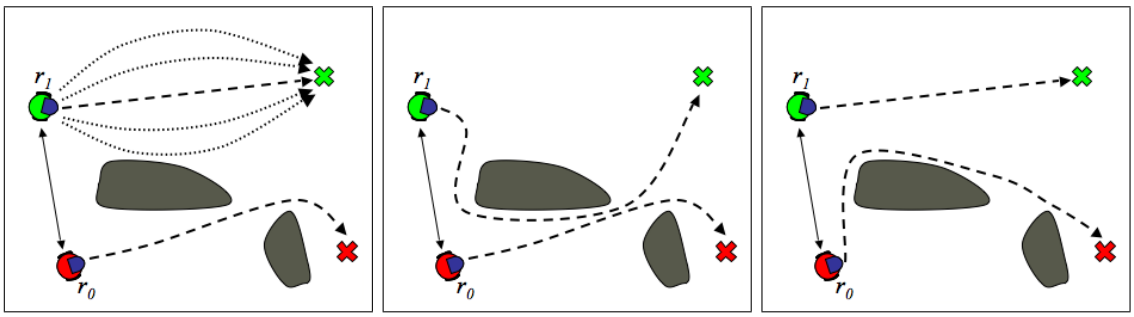
7.4 Design considerations: active and passive coordination

The next step to implementing LOSEX after formulating the problem is to design the Hoplites coordination mechanisms.

Passive coordination design

In LOSEX, each robot has a predetermined set of sites to visit and a predetermined order in which to visit them. Therefore, unlike in security sweep, a robot does not have flexibility in choosing its next goal: it must always plan to the next site in its tour. Its challenge is to choose a path to that site that maximizes connectivity with its uplink but minimizes distance.

An interesting property of LOSEX is that if a robot's current path p will not allow it to maintain line-of-contact with its uplink, then it is likely other paths that are similar



(a) (b) (c)

Figure 7.6: Robot r_0 and r_1 each have a site to visit; r_0 is r_1's uplink. r_1's shortest path to its goal (dashed) does not provide connectivity with r_0 and neither do other paths in the same homotopic class (dotted) *(a)*. Only a path in a different homotopic class offers connectivity.*(b)*

to p (specifically, those in the same *homotopic class*) will also not allow connectivity. For example, in Figure 7.6 *(a)*, robot r_0 and r_1 each have a site to visit and r_0 is r_1's uplink. Notice that r_1's shortest path to its goal (dashed) does not provide connectivity to r_0. Indeed, neither do other paths in the same homotopic class (dotted). Only the dashed path in Figure 7.6 *(b)*, which is of a different homotopic class, offers connectivity. Ideally, a robot would generate a set of candidate paths that includes one path from each homotopic class in the environment. However, generating this set in continuous space is infeasible: using deterministic algorithms such as A* is intractable (though, in theory, we could guarantee finding the complete set) and using less-expensive randomized algorithms such as RRTs cannot guarantee finding the set. Given these difficulties, we optimize on distance and computation instead and generate one candidate path in passive coordination that is the shortest path from the robot to its goal. We search for alternative paths only when it is necessary.

Choosing coordination strategies

If a robot's current path will cause it to lose connectivity with its uplink, its first and best course of action is to find a new uplink. This minimizes distance (since the robot's path should be the shortest path to its goal) and it is computationally inexpensive. To make swapping more effective, however, we only allow a robot to change to a new uplink when doing so will provided connectivity for at least some reasonable amount of time (otherwise, it would have to find another uplink immediately thereafter).

If this fails, the robot must either plan a new path or actively coordinate with its teammates. Because achieving connectivity can require a significant deviation in route, it is often unclear whether generating a new local path is less costly than actively coordinating

to change a teammate's path. In Figure 7.6 *(c)*, for example, robot r_1 could purchase r_0's participation in the plan that requires r_0 to change path homotopies because it is less expensive than r_1 changing its own path as in Figure 7.6 *(b)*. As we discuss later, we incorporate new local plan generation into active coordination to maximize flexibility.

Active coordination design

At this point a robot must select some teammate(s) with which to try and actively coordinate. As with security sweep, the structure of this problem makes it natural for a robot to actively coordinate with its current uplink, and this is the strategy we use. Unlike in security sweep, however, robots could potentially be assisted by other teammates; for example, teammates which are not currently in the robot's line of sight could be cheaply paid to move into view and provide connectivity. The tradeoff is that, in a large team, exploring all such options would be intractable. We believe exploring these options would be fruitful in real-world systems, but we omit it from our implementation for simplicity.

Planning team solutions

The next stage of active coordination is to produce a team solution. In security sweep, a robot r trying to actively coordinate with its teammate t proposes a solution for t. As we described in Chapter 4, however, this requires r to have a model of its teammate and information about its teammate's goals, constraints, and current plans. This approach is neither flexible since it requires the teammate to consider only the proposed actions, nor conducive to a heterogeneous team where it may be difficult to have models of teammates. In our implementation on LOSEX, we use a different approach. Rather than planning for its teammate, r shares a plan of actions it wishes to take (e.g. a path to its goal) and the period for which it wants assistance. Given that t has a model of its own behavior and accurate information about its plans and constraints, it is more likely than r to produce a good solution.

The teammate t has at its disposal implementations of many of the algorithms we described in Chapter 5. We use them in order of increasing complexity so that we may solve the problem with the simplest possible approach. The simplest is a coupled algorithm that uses prioritized planning to reduce problem complexity. Specifically, it first plans direct paths for each robot to its goals and then uses A* to synchronize them in time. In implementation, these direct paths were already computed during passive coordination, so this algorithm is very fast because it makes use of prior computation.

When this fails (as it often may because it does not change path homotopies), we use a domain-specific planner with relaxed constraints. Given our domain knowledge that LOSEX requires generating qualitatively different plans, we have developed the algorithm

that generates a new path for t in a homotopic class that is similar to the homotopic class of r's path. This algorithm is fast because it uses fast deterministic planners (e.g. A*); moreover, the resulting path is likely to offer connectivity with r. r_0's path in Figure 7.6 is an example of such a solution.

Lastly, if this also fails, we use a Rapidly Exploring Random Tree (RRT) [80] to generate an entirely new path for t. The algorithm grows a tree from t's current location to its next goal and discards any actions that would cause it to lose line-of-sight contact with r. In theory, the RRT can generate any solution that our previous algorithms could, but that is not guaranteed because of randomness. The benefit of RRTs is that they can find a number of approaches that other planners cannot. Nevertheless, they are not guaranteed to terminate as A* and other algorithms are, so we must artificially terminate after some set time.

The market mechanism

We use direct peer-to-peer negotiation to achieve active coordination. Continuing from our running example, if teammate t is able to find a plan p that would allow it to satisfy robot r's constraints, it sends to r the marginal cost of the p compared to its original plan. This is the cost for which r has to compensate t and can include distance and time costs as well as any violations it would create between t and its uplink. Notice that t does not share the details of p with r: r does not need to know how t is meeting its requirements, only that it is.

Once r receives a quote q from t, it must compare q against its best alternative. Recall that during passive coordination, robots only generate a single path to their goal. Since we have explored such a small region of our search space and since we know the resulting solution is costly in terms of constraint violations, it is unlikely that r's current path is its best alternative. Therefore, r now takes the opportunity to replan its own path. r uses the same planning techniques that t did to find a solution. This allows it maximum opportunity to find a better alternative. It then computes a reserve price using this alternative path; if q is less than the reserve price, t's solution is better than r's and r pays t for its participation in plan p. If, on the other hand, q is more than the reserve price, r rejects t's offer and now adopts its best alternative plan rather than its original plan. By searching a larger solution space during active coordination rather than passive coordination, r incurs the expensive computation only when it is necessary. Moreover, r's search can occur at the same time that t is searching for p (rather than first waiting for the quote q), and we can parallelize the computation and speed up the system.

7.5 Key experiment features

We use LOSEX to implement and evaluate several key features of Hoplites. We briefly mention them here and discuss them further with experiments in Section 7.6. First, we use a flexible team structure with uni-directional constraints to allow the team to rearrange itself to meet the needs of the mission. Empirically, we evaluate how often the team is able to solve the problem because of this flexibility and how this flexibility changes as we change the number of robots on the team.

Second, we use very flexible commitments to facilitate responsiveness to new information, especially when environments are unknown. Specifically, we allow a robot that is committed to providing assistance to a downlink to abandon that commitment whenever it results in constraint violations with its own uplink. Because constraints are uni-directional, the abandoned downlink will discover that the uplink's current solution does not provide connectivity and will try to re-initiate a new instance of active coordination. This will take into account all the new information that invalidated the previous attempt and should find a solution if one is possible using the methods available. We evaluate this flexible commitment strategy by assessing the impact of prior environment information on solution quality; if our commitment strategy is insufficient, we should expect a sharp degradation in performance.

Third, we provide our robots with a planning toolbox that enables them to use increasingly sophisticated algorithms as the problem scenario requires. We evaluate how often each algorithm is used in practice and how often they successfully find solutions. We also evaluate Hoplites's ability to scale to large teams and its sensitivity to different implementation parameters.

7.6 Simulation experiments and results

We have designed our experiments to evaluate the three most important claims of this thesis. First, that Hoplites significantly improves solutions by enabling robots to escape local minima by buying each other's participation in complex plans. Second, by selectively injecting these pockets of complex coordination, Hoplites can provide these improvements while remaining competitive with other distributed approaches. Third, this selective complexity allows Hoplites to outperform centralized approaches as well because it can often exploit planners with performance guarantees which a centralized approach cannot.

We have already demonstrated the strength of Hoplites via experiments in security sweep. Our implementation of security sweep, however, used a very complex form of active coordination in which robots could chain their coordination. Although chaining is a very powerful coordination feature, we also want to show that simpler implementations of

Hoplites can still provide significant performance improvements. To this end, our implementation of LOSEX uses very simple active coordination.

In our simulation experiments, we compare Hoplites to two other approaches and a "base" approach. In this section we discuss in detail these frameworks, outline our experiments, and present and discuss our results. We also experimentally explore the impact of a number of design decisions including the cost functions and flexible team structures.

Frameworks

We previously compared Hoplites to MVERT, P-MVERT and passive coordination on the security sweep domain. Having clearly demonstrated that our problem space requires both planning and tight coordination, we can omit both MVERT and P-MVERT from future comparisons as they do not provide these features. Instead, we compare Hoplites to passive coordination as the state-of-the-art distributed approach to the problem. We also compare Hoplites to a centralized approach to evaluate the impact of adaptive complexity. Finally, we include results from a base algorithm just to provide ground truth about robots' initial tours.

The Passive Coordination Framework. Our passive coordination implementation follows mostly from our discussion of design considerations in Section 7.4. A robot r selects as its default path the shortest path to its next target using A* and compares this to its uplink u's path. Recall that in the full Hoplites framework, if this path fails to maintain connectivity with u, r searches for its best alternative plan during its efforts to also actively coordinate with u. Obviously passive coordination does not involve r negotiating with u. For fairness, however, we do allow r to explore alternative paths that it could take to maintain connectivity with u. Specifically, r uses the same three planning algorithms available to robots during active coordination to search for an alternative plan. If such a plan is found, it uses the new plan; otherwise, it stays with its original plan. By giving passive coordination every planning opportunity available to Hoplites, our comparison will highlight precisely the impact of active coordination.

The Centralized RRT Framework A key property of Hoplites is that it is responsive to the complexity of the immediate problem. By comparing it to passive coordination, we are set to evaluate the impact of selectively injecting complexity in the approach. However, we are also interested in evaluating the impact of selectively injecting *simplicity* into the approach. In sum, these comparisons would allow us to answer the question, "What is the impact of adaptive coordination?"

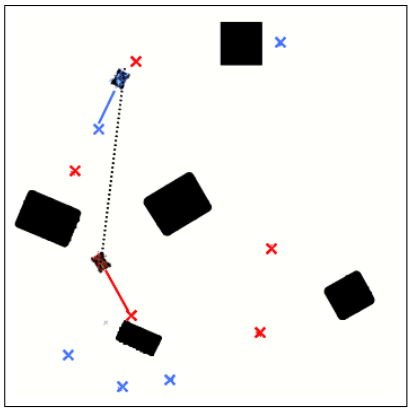


Figure 7.7: A snapshot from our simulator of a typical experiment with two robots and eight cities.

To this end, we have compared Hoplites to a centralized approach that essentially does active coordination among all of the robots all of the time. To make this tractable, we use an RRT algorithm based closely on Ferguson's work that solves the general constrained exploration problem [98, 99]. We have modified it to allow planning to a series of sites for each teammate: each call to the tree returns a path for the team that guides at least one robot to its next goal while keeping the others in communication contact. For fairness, we use the same algorithm parameters and path smoothing techniques provided for the RRT algorithm in active coordination.

Base Algorithm. We would also like to provide a measure of how difficult the problem is to begin with and how well each approach is able to improve the team's solution over the default strategy. This default strategy is for each robot to just follow its pre-computed tour of cities (which will be of minimum cost); we then evaluate how many constraint violations occur as they execute these tours without coordinating. We include this comparison where it augments our discussion of the other approaches.

Experimental Setup

We tested our approach in a graphical simulation using two to ten simulated robots tasked with solving LOSEX. For each team size we randomly generated forty environments of size 200×200 cells which each contain five obstacles that range in size from 5×5 cells to 30×30 cells and interfere with communication. Figure 7.7 shows a typical environment.

Each robot is tasked with visiting four randomly generated cities while maintaining communication constraints with an uplink. When a robot has visited all of its cities, it does not need to maintain this constraint. (This prevents robots from aimlessly following their uplinks around just to maintain communication).

We experiment with giving the robots both perfect prior information about the environment and no prior information about the environment. We also impose a minimum planning horizon of 50 steps; that is, a robot must have planned a tour that is at least 50 steps but that may be longer since robots end each path at a particular city. However, for tractability and because teammates' paths frequently change, robots are only concerned with maintaining network connectivity for 10 steps at time. Robots reevaluated their connectivity every 3 seconds. Robots move at a speed of 5 cells per second and are equipped with simulated 360° laser scanners with a range of 200 units to provide sensory information as they move. In the distributed approaches, each robot ran as a separate software agent. We used shared memory between the agents and our simulator for regular high-frequency information such as position, environment, and path information to reduce the network burden on a single machine. However, robots used UDP for all messages relating to active coordination to allow for delays that might affect contracts and agreements on a real system.

Our global cost function C is exactly as in Equation 7.5, where each constraint violation has a weight of 100 while each unit of distance traveled and each second of time traveled as a weight of 1. Because robots must travel to specific cities in a particular order, we do not need to have a profit function with a reward as we did with security sweep; the same solution can be reached by using a local cost c function as in Equation 7.6. We include them again here for easy reference.

$$C = A\,T + B \sum_{r \in R} Distance(r) + \Gamma \sum_{t=0}^{T} \sum_{r \in R} Disconn(t, r, N(r))$$

$$c(a, r_i, t) = \begin{cases} \alpha\, t + \beta\, Distance(a) & \text{if } i = 0 \\ \alpha\, t + \beta\, Distance(a) + \gamma\, Disconn(r_i, n(r_i)) & \text{otherwise} \end{cases}$$

Lastly, we ran our experiments on a PowerMac G5 with Quad 2.5GHz processors and 1 GB of memory. Figures 7.21 and 7.22 at the end of the chapter show the results of an experiment formulated in this way with five robots and twenty cities; these results are presented as a set of 18 annotated snapshots that run the length of the trial.

Communication Requirements

The communication requirements of Hoplites on LOSEX are very similar to those for security sweep. Although we use shared memory in our actual system, the analytical results we include in Table 7.1 are communication requirements for a distributed robotic team. We measure the frequency and size of different types of communication. We provide candidate data about a team of 5 robots, though our experiments span team sizes of two to ten.

Suppose that R is the number of robots on the team. Since each robot only requires

	State	Env	Plans	Routes	Act Req	Act Rep	Active Acc
Msg Freq	R	R	–	–	–	–	–
Actual Freq	25	25	1	–	–	–	
Msg Size	$O(1)$	–	$O(\frac{sqrt(An)}{n})$	$O(log(R))$	$O(\frac{sqrt(An)}{n})$	$O(1)$	$O(1)$
Actual Size	c	–	$77 \cdot c$	0.7c	$77 \cdot c$	c	c

Table 7.1: Communication requirements of Hoplites on LOSEX. R is the number of robots on the team, T is the number of teammates with which each robot shares information, L is the planning horizon, P is the replanning period, and c represents some constant.

state information from its one uplink, the number of state messages is R at each time step. Note that although there are a constant number of messages, these R may not be evenly distributed among the team: one robot may have several downlinks requiring information while another robot may have none. Specifically, each of five robots moves at a frequency of five cells per second and each requires information from at most one of its teammates, resulting in 25 state messages sent per second. The size of state messages is constant: each robot shares an (x, y) position and its point in the current plan.

Secondly, robots need to share environmental information to anticipate and avoid losses in line of sight. If each robot receives new sensory information at each step and shares this information with its downlink, this results in R environment messages per second. In practice we want robots to communicate only when they have new information (which decreases communication requirements), and we may want them to aggregate this information from other teammates so that each robot can have a larger picture of the environment (which increases communication requirements). We use this latter feature in our implementation: when a robot shares environmental information with its downlink, it also includes map information recently received from other teammates. Moreover, the size of these messages can vary depending on the complexity and size of the environment and the type of sensors.

Third, in our system, robots communicate their plans only when they change; however, unlike with security sweep, robots do not replan at a fixed frequency. Instead, they replan whenever the length of the current tour drops below the lookahead distance L (50 steps in this case) or when they expect constraint violations with their teammates. Thus, we cannot evaluate this analytically. The length of paths is also not constant since our lookahead L is a lower bound. However, by evaluating the expected distance between cities, we can compute an expected length of the shortest path to the next city. The expected length l of an optimal TSP tour of n cities in an environment of area A can be approximated by

$$l = 0.765\sqrt{(n * A)} \tag{7.7}$$

where, in our case, $n = 4$ and $A = 40,000$ [106]. Thus, $l = 306$ (which we find is consistent with our experimental results). From this, we can say that the average distance between adjacent cities is $300 \div 4 = 76.5$ and thus require a path of 77 (x, y) positions.

The communication frequency of route requests and active coordination cannot be determined before-hand; it depends upon the complexity of the problem. From our results, however, we find that in a team of five robots there are 12.2 route requests, 8.6 attempts at active coordination, and 2.0 successful instances of active coordination. For each attempt at active coordination, one request is sent out that contains plan information required by the robot considering the proposal; this message size is also derived from the expected length of a tour. The reply to these messages consists of a single numerical bid and then response to that bid is another constant-size accept or reject message.

Results and Discussion

Base Experiments.

Our first set of experiments evaluates the ability to solve LOSEX given perfect prior information about the environment. Each graph presents results from passive coordination, Hoplites, and the RRT algorithm, grouped according to team size. Here, we just compare general trends between the approaches; we review trends across team size in our discussion of scalability later in the section. Figure 7.8 *(a)* shows the average number of constraint violation seconds incurred by the team. Hoplites is significantly better at satisfying the team constraints than passive coordination across all team sizes. Notice, however, that Hoplites cannot solve constraint violations as well as the centralized RRT: for teams of two and three robots, there are no constraint violations and the RRT is able to maintain line of sight perfectly. This is not surprising since we expect that having team-wide information and being able to plan for every teammate at once will result in a greater ability to satisfy constraints.

However, for teams of five and ten robots RRTs are unable to find a centralized solution in a reasonable time (we stopped any algorithm after 180 seconds without a solution) and we have no data for RRTs for these larger team sizes. Although we expect RRTs to eventually be unable to handle large team sizes, it is at first glance surprising since Ferguson et. al. [98] produced solutions for teams of up to 10 robots in about 20 seconds. Closer inspection reveals that having multiple cities significantly complicates the problem though it is an apparently small domain change. In Ferguson's work, each robot had one goal, and these goal positions were fully connected by line of sight, and all were equally distant from the robots' start positions. Thus, the RRT could plan using a strong heuristic biased towards those goal locations and did not have to deal with robots arriving at their goals significantly earlier than their teammates. In contrast, in our problem, robots' cities are

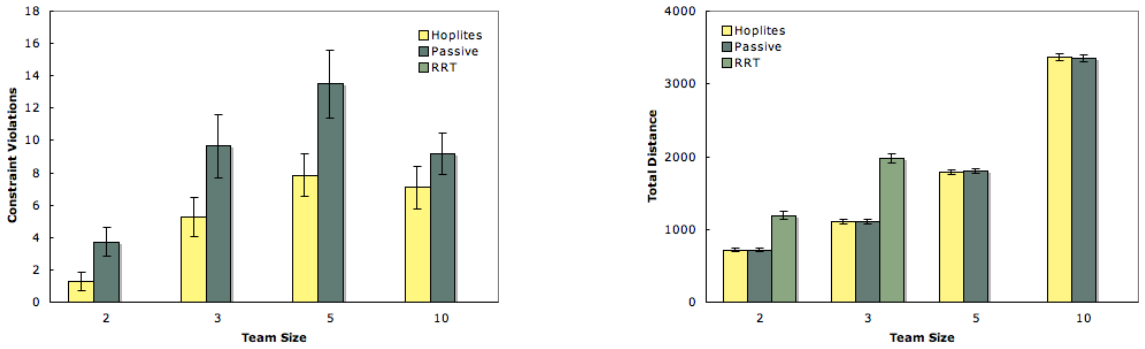


Figure 7.8: The performances of each approach in terms of the average number of violations *(a)* and the average distance traveled by a robot *(b)*. Error bars indicate the standard errors of the means.

not co-visible and are not equally distant from previous cities in their tours, so our RRT does not have a well-defined goal and it must discover the goal using an unfocussed search. This adds subtle but incredible complexity to the problem and makes it intractable for even medium sized teams. Moreover, the impact of this increase in problem complexity is not specific to RRTs: we expect the same qualitative result with a deterministic algorithms such as A*. Therefore, no optimal solution is possible even for the two robot case.

Figure 7.8 *(a)* suggests that Hoplites is better than passive coordination and that the RRT is better than Hoplites (for small team sizes) at maintaining constraints. Nevertheless, to evaluate overall solutions, we must consider the overall ability to balance competing needs of maintaining constraints and minimizing distance and mission time. Figure 7.8 *(b)* shows the total distance traveled by the team. From our base tour computations, we find that Hoplites and passive coordination both increase the distance of the robots' tours by between 5 and 10% over the their original tours across all team sizes. Moreover, Hoplites and passive coordination produce equivalent solutions in terms of the distance traveled by the team. Figure 7.9 *(a)* presents the overall mission time. This measure does not include planning time because, as we explained in Chapter 6, planning time is hardware dependent while mission time is algorithm dependent. Here again, Hoplites and passive coordination perform about the same, with Hoplites missions requiring about 6% longer to complete than passive coordination. This extra time is due to robots paying their uplinks to synchronize their paths and wait; this does not add distance and is a cheap way to maintain constraints. Compared to the minimum mission time, passive coordination adds 15% more time and Hoplites adds 25% more time.

These results have significant implications. In Hoplites, when a robot is faced with a communication break with its uplink, it is able to use active coordination to vet a greater variety of solutions to the constraint satisfaction problem over larger portions of the team

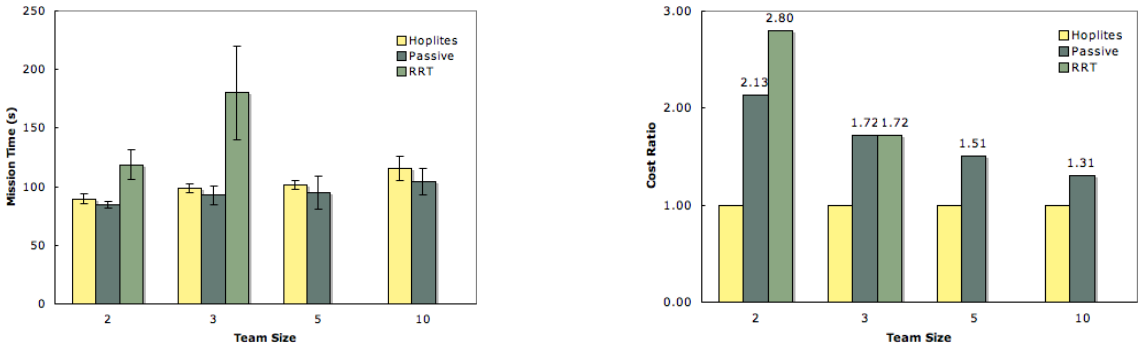


Figure 7.9: The performances of each approach in terms of the average mission time *(a)* and the overall solution cost ratios where Hoplites is normalized to 1 *(b)*. Error bars indicate the standard errors of the means.

than passive coordination. Firstly, this means that Hoplites can find solutions when passive coordination cannot (which results in a decrease in constraint violations in Figure 7.8 *(a)*). Secondly, it also allows Hoplites to find *better* solutions than passive coordination to the same problem scenarios. In sum, Hoplites is able to maintain constraints significantly better than passive coordination, without compromising the distance traveled by the team or the length of the mission.

Let us also return to the RRT approach to consider its ability to trade off constraints against mission length and distance. Figures 7.8 *(b)* and 7.9 *(a)*) show that RRTs actually have *greater* distance and mission time requirements than either of the other approaches. The strength of RRTs is that they can find feasible solutions when other approaches fail because they explore rapidly and without regard to cost. The drawback which we observe here is that the solutions they find are not optimized on other factors. In this case, the solution to the problem of getting robots to goals without violating communication constraints is not optimized on time and distance. Moreover, while some new approaches have looked at incorporating cost into the search [99], they require even more planning time to produce good solutions than our current RRT which is already only feasible for very small team sizes.

In sum, active coordination allows Hoplites to find much better solutions than *both* the distributed passive coordination and the centralized RRT. To formally explore the overall solutions, we compute the average solution cost over all our approaches using Equation 7.5. Recall that there is a fixed minimum cost incurred by each approach in order to complete the mission; this is the sum of distance costs of the robots' optimal tours and the cost of the mission time required by these tours. Because these are outside of the control of our approaches, we subtract these costs from the original solution cost and are left with a more accurate measure by which to compare the approaches. Figure 7.9 *(b)* presents the relative

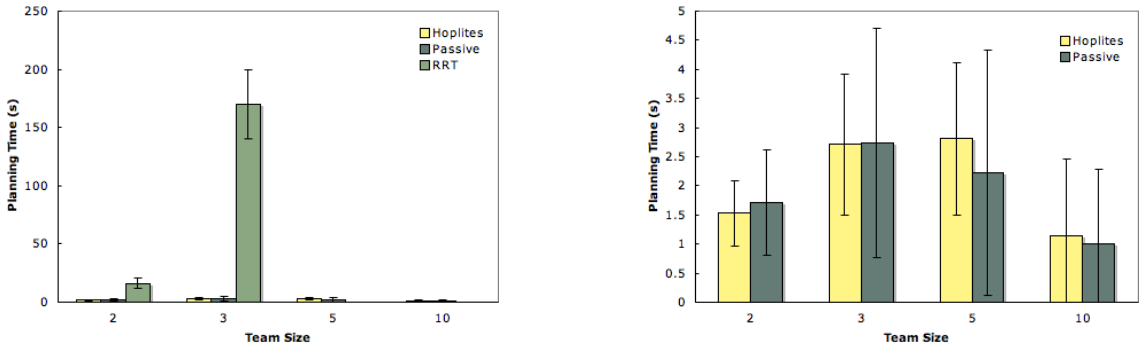


Figure 7.10: The performances of each approach in terms of the average planning time per robot. In the RRT approach, there is only one agent planning for the team. Error bars indicate the standard errors of the means.

costs of the different approaches when normalized by Hoplites's performance The numbers above each bar are the actual ratio of each approach's cost to Hoplites's cost.

That Hoplites significantly outperforms passive coordination clearly verifies our first claim that it improves solutions by enabling robots to escape local minima by buying each other's participation in complex plans. Moreover, that Hoplites *also* outperforms RRTs corroborates our claim that by selectively injecting complexity, it can produce better solutions than even centralized approaches. That is, RRTs are not adaptive and always produce feasible but highly suboptimal solutions. Hoplites, on the other hand, selectively applies these more sophisticated algorithms and thus pays the price of these approaches only when absolutely necessary. Much of the time, it is able to exploit simple planners with solution guarantees and thus provide better solutions overall.

Finally, we must evaluate our claim that Hoplites can provide these high-quality solutions while remaining competitive with other distributed approaches. Figure 7.10 *(a)* shows the planning time for each approach. This is the total planning time per robot summed over the entire experiment. Because RRTs are centralized, this is the planning time for the single planning agent. First, the planning time for RRTs is several orders of magnitude worse than either passive coordination or Hoplites. Figure 7.10 *(b)* omits the RRT values to focus in on the difference between passive coordination and Hoplites. We see that Hoplites has essentially the same planning requirements of passive coordination. Hoplites consumes more time per planning iteration but actually plans less frequently because it finds solutions using active coordination. Meanwhile, passive coordination consumes less planning time but has to replan more frequently as robots must often try (in vain) to meet constraints without the active assistance of their teammates.

The overall conclusion is that, by intelligently and selectively incorporating complexity into coordination, Hoplites provides large improvements in solution cost, essentially for free.

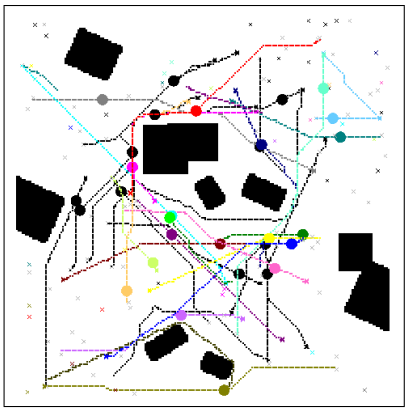


Figure 7.11: A snapshot from our simulator of a team of thirty robots and 120 cities. Circles represent robots, black areas represent communication obstacles, x's mark cities the team must visit, and dotted lines indicate robots' paths. For clarity, communication links between robots are omitted.

Scalability

We have provided experimental results on team sizes from 2 to 10 robots to evaluate Hoplites's ability to scale to large teams and the overall impact of increasing team size. Our comparison focuses on the difference between passive coordination, which is fully distributed, and Hoplites. We have already shown that the RRT approach does not scale. Specifically, from Figure 7.10, we can see that the RRT's planning time appears to grow exponentially in the number of robots, which is what we would expect from a centralized planner that plans in the problem's full configuration space. In comparing Hoplites and passive coordination, we find that not only do they have the same planning requirements, but the planning time per robot grows roughly linearly (between 2 and 5 robots, inclusive).

First, this confirms that Hoplites can scale to large teams. Second, notice, that the planning time *decreases* as we increase our team size to 10 robots. This is because the problem complexity of LOSEX decreases when we add more teammates, which we touched upon briefly in Chapter 4. That is, with more teammates, it is easier for robots to avoid violating team constraints (e.g. by swapping uplinks) which, in turn, reduces the amount of planning required. Indeed, if we return to the graph of constraint violations versus team size in Figure 7.8 *(a)*, we can see that the total number of constraint violations actually decreases from five to ten robots because the problem has become easier. Thus, we cannot judge the complexity of the problem in advance of execution, and we must use adaptive complexity to solve these problems efficiently.

Lastly, we have experimented in simulation with teams of up to thirty robots as shown in Figure 7.11. Figure 7.20 (at the end of the chapter) shows snapshots from an experiment with 20 robots and explicitly illustrates simultaneous pockets of active coordination within

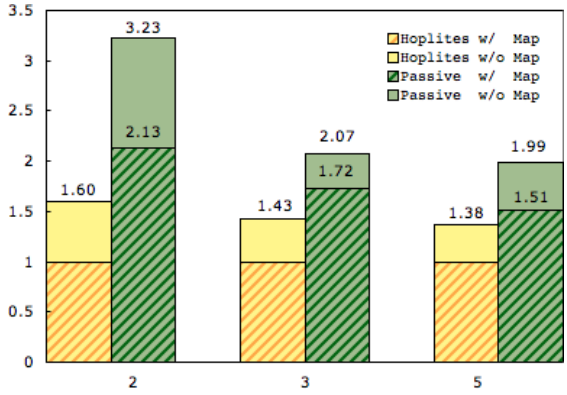


Figure 7.12: The performances of passive coordination and Hoplites with and without prior maps. For each team size, we normalized Hoplites performance with a prior map to 1 and present the solutions of the other approaches relative to this value.

a large team.

Prior information

Thus far, we have given our teams perfect information about their environments, but this is not a realistic representation of the real world. Moreover, in tasks such as exploration on which our work is based, unknown worlds are part of the problem definition. Although we showed that Hoplites successfully operates without prior information in our security sweep experiments, here we formally asses the impact of the prior map.

Figure 7.12 presents the performances of passive coordination and of Hoplites with and without prior maps. For each team size, we normalized Hoplites's performance with a prior map and present the performances of the other approaches with and without prior maps relative to this value. The performance clearly degrades overall when we remove prior maps for both Hoplites and passive coordination. Still, Hoplites's performance without the prior map is across the board significantly better than passive coordination's performance without the map.

Moreover, a closer look at the details presents further interesting trends. Hoplites' performance without a prior map degraded more than passive coordination's performance without the map, respective to their individual performances with the map. We expect this because active coordination is sensitive to prior information: it is less flexible than passive coordination because teammates commit to providing service to their teammates. More importantly, despite this degradation, Hoplites' performance without a prior map is better than passive coordination *with the map*. Clearly, Hoplites is able to operate successfully without prior information and it is the stronger framework regardless of the presence of prior information.

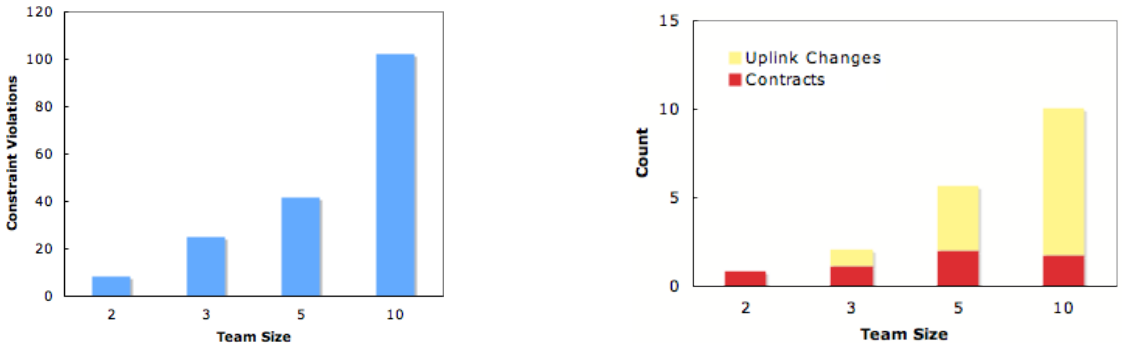


Figure 7.13: The number of constraint violations when there is no coordination and when they cannot swap links *(a)*. The average number of contracts formed compared to the average number of uplink swaps *(b)*.

Team Structure

In our initial development of the problem in Section 7.1, we claimed that a flexible team structure plays an important role in a team's ability to solve the problem. Here, we use results from our experiments with a team of five robots to evaluate the importance of flexibility empirically. Figure 7.13 *(a)* shows the base number of constraint violations when there is no coordination between teammates and when they are not able to swap links; clearly constraint violations increase significantly with team size. Yet, recall from Figure 7.8*(a)* that the number of constraint violations eventually decreased for our approaches which used swapping. We can explain this if we examine Figure 7.13 *(b)*, which compares the average number of contracts formed between teammates to the average number of uplink changes that occur during an experiment, categorized by team size. With two robots, swapping of uplinks is obviously not possible. As we increase the number of robots, however, swapping occurs increasingly often; indeed, several times more often than active coordination. This clearly suggests that a flexible team structure plays an important role in solving the task and that a fixed structure would result in very poor solutions.

The planning toolbox

Hoplites decouples the planning algorithms available to the robots from the coordination mechanisms they use. This allows robots to choose planners according to the difficulty of the problem scenario rather than the complexity of the coordination mechanism. Moreover, when a simple planner fails to find a solution, they can use a more complex planner and thus attack problems with increasingly sophisticated algorithms.

The simulation snapshots in Figure 7.14 demonstrate how our four different planning algorithms can be used in the same experiment to provide different solutions. In Figure 7.14

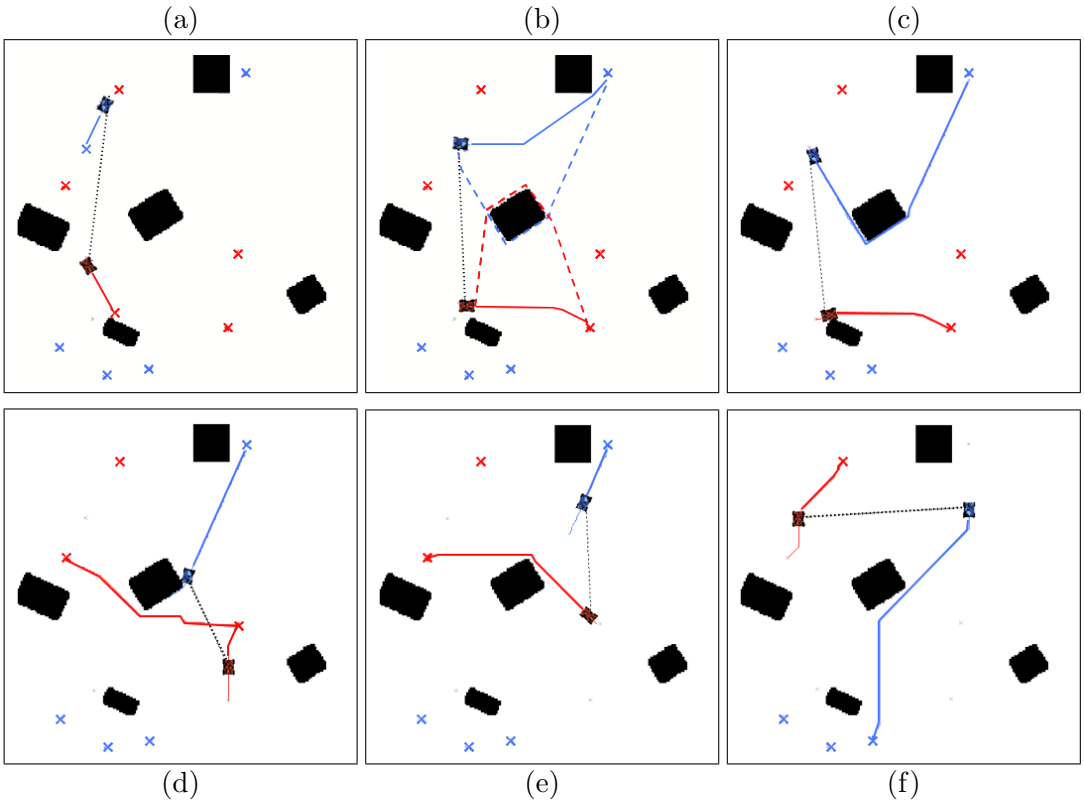


Figure 7.14: Snapshots from a simulation involving two robots and a series of sequential goals to be visited by each robot. This experiment demonstrates the range of planning techniques.

(a), the robots use A* to find direct paths; this is the simplest algorithm and provides very efficient solutions in simple scenarios. In Figure 7.14 *(b)*, r_1 tries to actively coordinate with r_0; the dotted lines are the two alternative paths developed by the robots using our domain-specific planner. After comparing costs, the better solution is for r_0 to change its path, as shown in figure 7.14*(c)*. Then, in Figure 7.14 *(d)*, r_1 uses the most complex planner, an RRT, to find a feasible path for actively coordinating with r_0 when simpler planners cannot find a solution. In Figure 7.14(e), r_0 uses the domain-specific planner again, this time to find an independent path when active coordination with r_1 fails. Finally, in Figure 7.14(f) r_0 actively coordinates with r_1 by using a prioritized planner to synchronize its path to r_1's without changing its route. As illustrated in this example, a large planning toolbox increases the chances of finding a solution, while also encouraging the use of less expensive planners whenever possible and the use of complex planners when necessary.

We have also quantitatively evaluated the impact of the different algorithms in our planning toolbox. For a team of three robots, we monitored the number of times each planning algorithm was used and how often each was successful. The results are presented

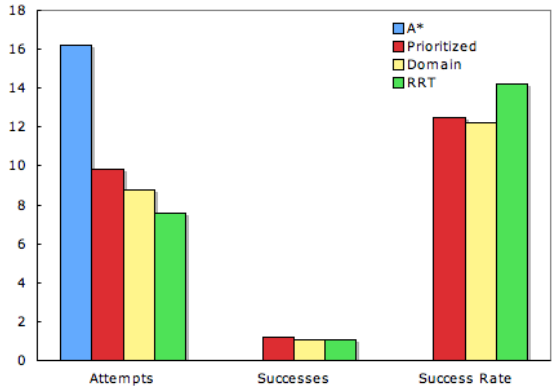


Figure 7.15: Comparison of different planning algorithms. We measure the average number of attempts , the average number of successes, and the percent rate of success for each approach. A* was successful in finding a path 100% of the time; these results are excluded to avoid skewing the graph.

in Figure 7.15. First, notice that we use the simplest planner A* most frequently to find direct paths between robots and their goals. A* found solutions 100% of the time (not shown), but these solutions sometimes did not maintain communication constraints. Then, robots attempted active coordination in which the prioritized planner, the domain specific planner, and the RRT were used in a cascading manner. That is, when the first one failed to find a solution that maintained constraints, the next more-sophisticated algorithm was used. The success rate of each algorithm was about 13% and the overall success rate was 35%. Clearly, each of these three algorithms contributed equally to the solution and removing any single algorithm would have resulted in an increase in solution cost. This demonstrates the overall benefit of a planning toolbox with several different approaches.

Sensitivity to function weights.

A perceived drawback to market-based approaches is that the performance of the system is very sensitive to the weights in the cost functions, and that, if these weights are not chosen perfectly, the system will fall apart. We can evaluate the validity of this claim empirically with our implementation on LOSEX.

Specifically, we want to examine how changing the relative importance of minimizing resources and maintaining constraints in the robots' local cost functions impacts the solutions produced. Let us begin by considering the impact of the weight change in principle.

As we discussed in Chapter 4, robots' reserve prices reflect how much they are willing to pay their teammates to take actions to satisfy constraints that, without their assistance, would not be maintained. Now, as we increase the importance of constraints, robots' reserve prices will increase and they will be willing to pay a higher price for teammates'

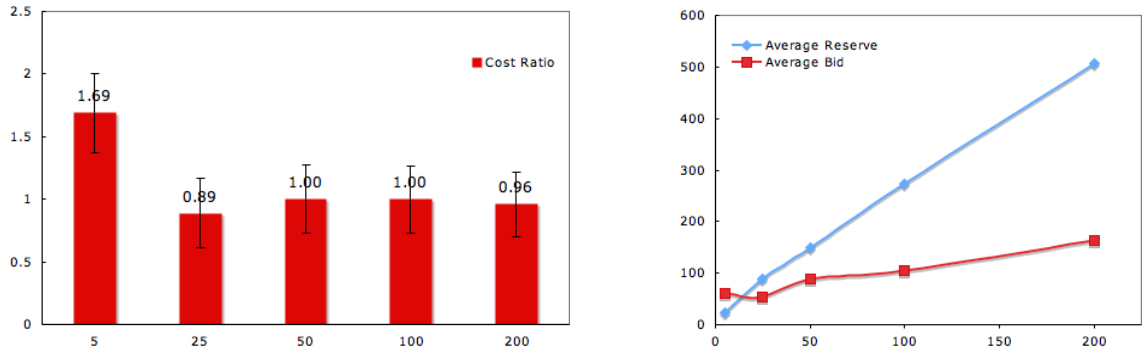


Figure 7.16: These graphs show the average cost ratio *(a)* and the average reserve and bid prices *(b)*.

participation. Simultaneously, the teammates' bid prices will increase because the cost of losing connectivity with their own uplinks has increased. However, increasing the importance of constraints will increase the average bid price less than it will increase in average reserve price. This is because, unlike reserves which always include the cost of constraint violations, only some teammate prices involve constraint violations costs; those that do not will be immune to changes in constraint weights. This should result in fewer constraint violations and a larger expenditure of team resources. Finally, because we can vary gamma continuously, we can expect the solution cost to vary continuously as well; this should hold for an arbitrary domain and not just our problem or our implementation.

We evaluated this hypothesis with a simulated team of three robots. The weights of the distance and mission time components in Equation 7.6 remained constant across experiments ($\alpha = \beta = 1$, as before) but the weight γ of constraint violations varied; we experimented with $\gamma = \{5, 25, 50, 100, 200\}$. We compared the solutions by evaluating them with the same global cost function; specifically we use $A = B = 1$ and $\Gamma = 100$ as we have with all previous experiments.

Figure 7.16 *(a)* shows average the cost ratio of the solutions generated by each local cost function, normalized to the results from $\gamma = 100$ (the local cost function analogous to the global cost function). Even significantly changing the local utility function by doubling or quartering γ has no effect on the solution. Indeed, we only get significantly different costs when we place almost no importance on maintaining constraints ($\gamma = 5$). Closer inspection further reveals that the solutions from $\gamma = 25$ to $\gamma = 200$ are the same at component level as well: they have the same mean number of constraint violations, the same mean distance traveled by the team, and the same mean mission time. Thus, we can conclude that these approaches are actually producing the exact same solutions. Only $\gamma = 5$ produces different solutions where constraint violations increase and distance and mission time decrease. Overall, we can say that the system is insensitive to the weights

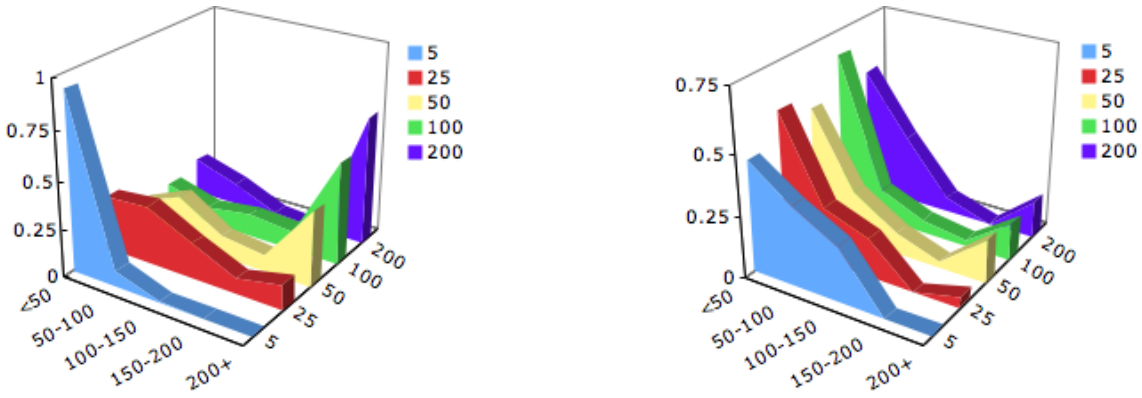


Figure 7.17: These graphs show the distribution of reserve prices *(a)* and the distribution of bid prices *(b)* for different values of γ.

chosen, which completely contradicts our hypothesis.

To explain this phenomenon, we can further examine the change in the average reserve prices and bid prices as shown in Figure 7.16 *(b)*. From this data, we can see that they are consistent with our hypothesis! Both reserve and bid prices change, but the former changes more than the latter.

The apparent inconsistency is revealed if we look at the *distribution* of reserve and bid prices, not just the averages. Figure 7.17 shows the fraction of reserve prices that fall within the ranges $< 50, 50-100, 100-150, 150-200, > 200$ for each value of γ. By comparing these distributions with the average values, we can see that reserve prices are evenly distributed about the mean. The bid prices, however are *not* distributed evenly about the mean; Figure 7.17 shows that they they are heavily skewed towards smaller values for all γ. This means that almost all the solutions found by teammates do not involve constraint violations with their uplinks and therefore have low cost.

This makes sense when we consider the nature of the environment and how we generate solutions. First, the most likely reason that two paths lose line of sight is that they belong to different homotopies; that is they travel on different sides of obstacles. Because obstacles are large, two paths with any constraint violations are likely to have many. Thus, we have a somewhat binary world where either solutions maintain constraints completely or not at all. This means that our planning algorithms will also return extreme costs (i.e. either very low or very high) which translate into extreme bid prices. Furthermore, high bids are extremely rare because they reflect difficult situations for which there is often no solution. Indeed, for all values of γ, teammates find solutions for the robot requesting assistance only about 30% of the time; the remaining 70% of the time, no bid price is submitted.

Now that we understand this complex phenomenon of how reserve and bid prices are generated and distributed, it is easy to explain why changing γ has no impact on solutions

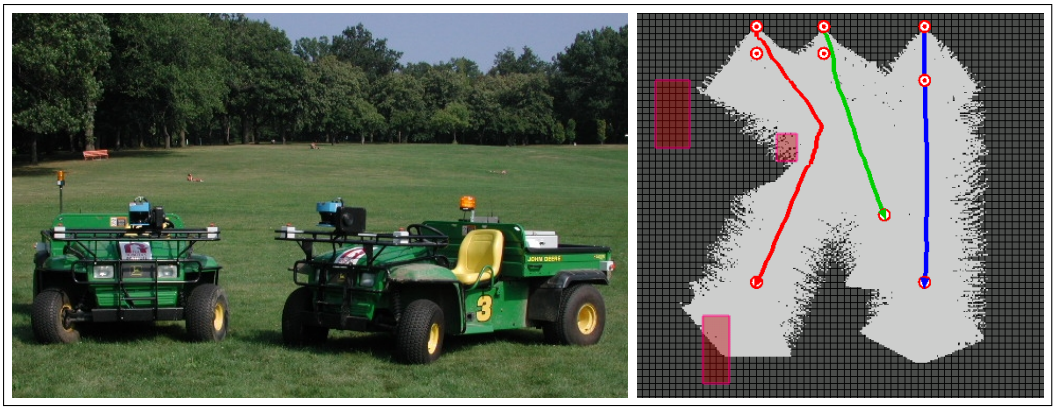


Figure 7.18: (left) Two of the three E-Gators used for field testing. (right) The results of a sample CE mission through an outdoor environment using the E-Gators in which the communication antenna is on board the left-most robot. Notice that the leftmost vehicle takes a path to the right of the center obstacle as a result of a contract made with the center vehicle.

except in extreme cases. At $\gamma = 25$, 65% of all solutions proposed by teammates are accepted (their bid prices are lower than the respective reserve prices). Furthermore, 90% of the failed solutions have bid prices in the highest bracket (> 200); this implies that they create more constraint violations than they solve and therefore they will *never* be accepted, regardless of the importance of constraint violations. Therefore, when $\gamma = 25$, robots have essentially already incorporated all possible feasible solutions and thus increasing γ has no effect (this 65% acceptance rate is constant for all values of γ we tested). Indeed we could increase γ to a million but that would not reduce the constraint violations because changing weights cannot create new solutions. It is only when the average bid price is significantly *higher* than the reserve price (when $\gamma = 5$) that we see a change in solution quality. In this case, only the very cheapest solutions are accepted and the solution acceptance rate drops to 3%. Essentially teammates must ignore constraints before solutions change.

On a final note, this does not necessarily mean that Hoplites is robust; it may be a product of the planning algorithms and the environment. Yet, given that constrained exploration is a general problem and that we are using general planning techniques, we believe that this insensitivity to weights will be common to other systems as well.

7.7 Validation on outdoor vehicles

We have also implemented Hoplites for LOSEX on a team of John Deere E-Gator robotic platforms designed for outdoor environments. These vehicles are equipped with inertial measurement units and GPS for position estimation, laser range finders for terrain observation, and wireless Ethernet for communication. Two of these platforms can be seen on

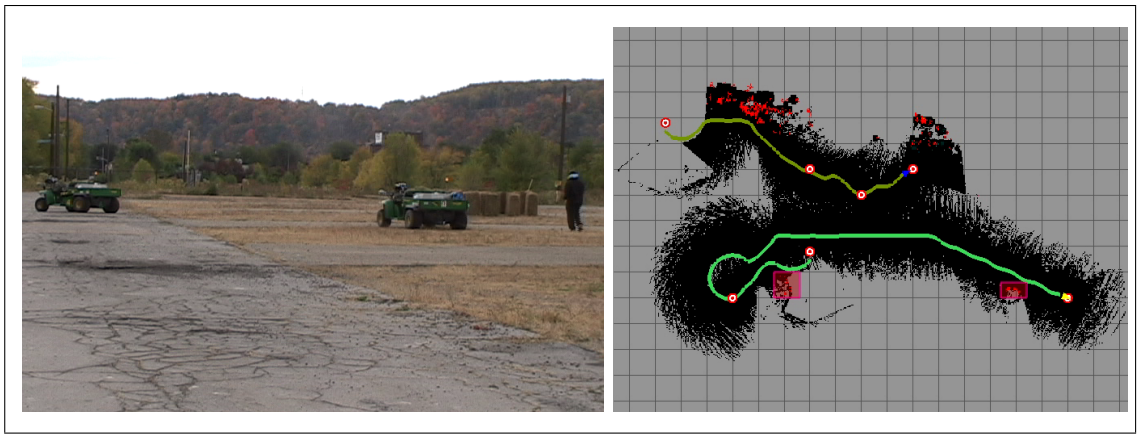


Figure 7.19: Two outdoor vehicles performing LOSEX.

the left in Figure 7.18.

Our implementation of Hoplites on these platforms follows directly from our implementation in simulation. Minor changes include that we use a meter resolution in our maps and that robots exchange state information with each other once per second. As with our simulations, we experiment with both known and unknown environments and use a series of goals for our robots.

In our first experiment, we gave each of our three robots two to three goal cities in a 60×80 meter environment; we provided a prior map of virtual communication obstacles but no information about navigation obstacles (e.g. rocks). The robots generated initial paths for the vehicles and updated these paths as new information was received. The right half of Figure 7.18 shows the results of this traverse. Targets represent the vehicles' initial and goal positions, shaded rectangles represent communication obstacles, dark areas indicate unknown terrain, and white areas represent traversable terrain detected by the robots' lasers. Here, a communication obstacle between the left and center robots required that they carefully coordinate to avoid losing line-of-sight contact. These two robots negotiated between two competing joint plans, determined that the best solution was for the leftmost vehicle to alter its course, formed a contract, and successfully executed the plan.

In a second experiment, we used two robots to solve LOSEX in a larger, 60×150 meter environment with no prior map information. As shown in Figure 7.19, the uplink robot (on the bottom) had two target sites and the downlink (on the top) had three target sites. Targets represent the vehicles' initial and goal positions, shaded rectangles represent the locations of the hay bales which acted as communication obstacles, grey ares indicate unknown terrain, black areas represent traversable terrain detected by the robots' lasers, and red rectangular areas represent the obstacle area detected by the lasers. Here, the uplink robot sensed the obstacles in the environment and shared this information with the

downlink robot which determined that a line of sight violation would occur. To solve the problem, the downlink robot actively coordinated with the uplink and paid it to make a difficult turn around its first target and travel around the far sides of the hay bales. As a result, they were able to visit their targets within the communication constraints of the problem.

7.8 Summary

In this chapter, we described our implementation of Hoplites on the constrained exploration domain and outlined in detail a number of design considerations. This domain highlighted a few features in particular: the use of a flexible team structure, a large planning toolbox, and uni-directional constraints between teammates.

Our experiments showed that Hoplites outperforms both centralized and distributed approaches on this problem by exploiting simplicity whenever possible and selectively incorporating complexity when necessary. For example, for a team of three robots, Hoplites outperforms both approaches by a factor of 1.72. For a team of five robots (for which no results were possible with the centralized RRT), Hoplites outperforms passive coordination by a factor of 1.51. Moreover, Hoplites provides these improvements with no increase in planning time.

These experiments also demonstrated a number of additional features. First, we show that the computation required by Hoplites scales linearly in the number of robots and is equivalent to the requirements of passive coordination. Second, Hoplites operates successfully under uncertainty; specifically, Hoplites finds better solutions without a prior map than passive coordination does *with* a prior map. Third, we show that the flexible team structure plays a critical role in the team's ability to solve problems. This flexibility causes solution quality to improve as we add more robots, whereas,without flexibility, it steadily degrades. Fourth, we demonstrate that our implementation is robust to changes in the problem parameters. Specifically, quartering or doubling the importance of constraints relative to the importance of completing the mission and minimizing the distance traveled has no effect on the resulting solutions; solutions are only affected when we almost eliminate constraints entirely. Lastly, we also validated Hoplites on a team of large outdoor robots solving two different scenarios of constrained exploration.

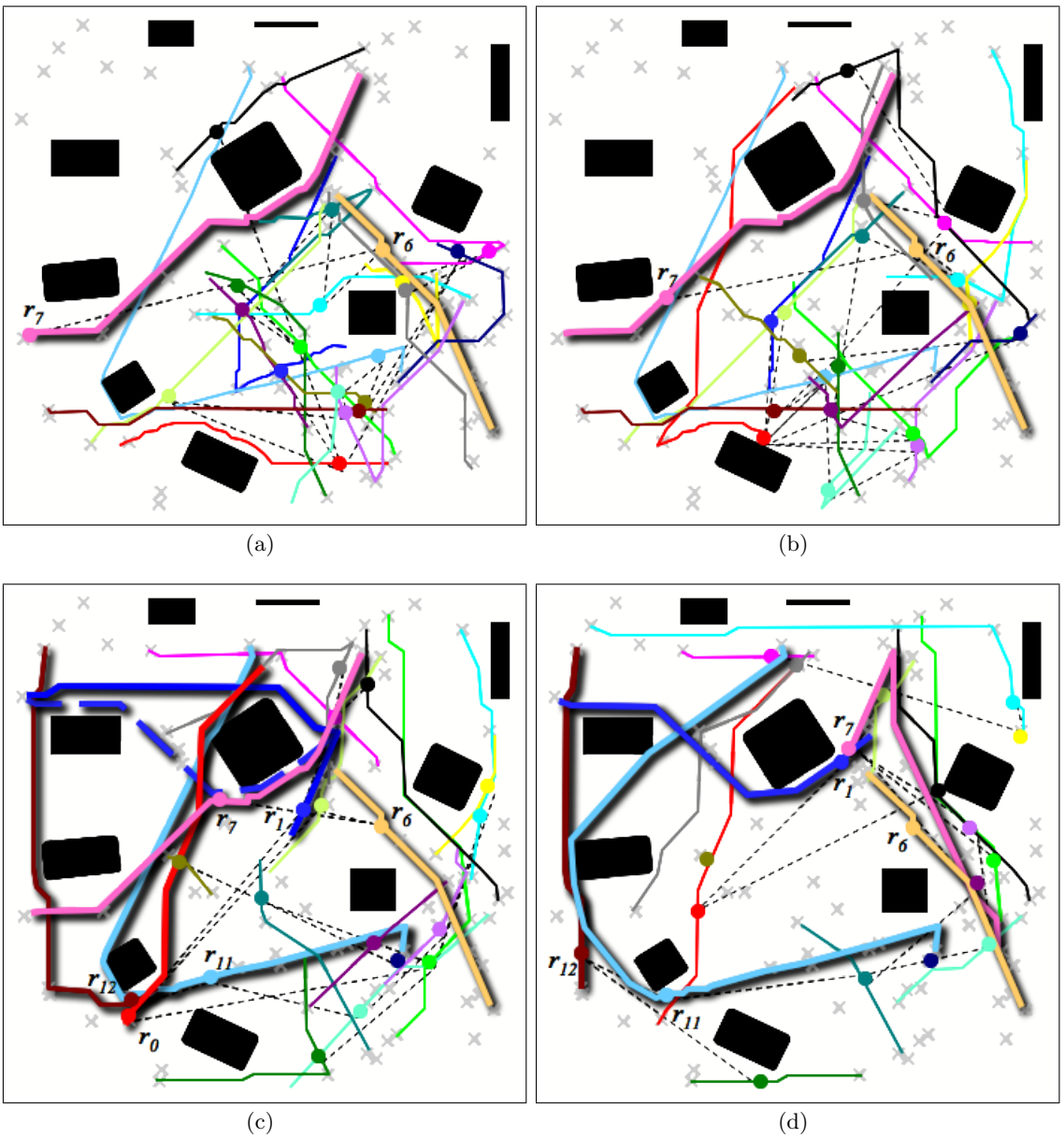


Figure 7.20: Four snapshots from a LOSEX simulation experiment with twenty robots and eighty cities. Circles represent robots, black rectangles represent communication obstacles, gray x's mark cities the team must visit, solid lines indicate robots' paths, and dashed lines indicate communication links between uplinks and downlinks. Subfigures *(a)* and *(b)* show robots r_6 and r_7 actively coordinating: r_7 pays r_6 to provide communication connectivity by waiting at its current location until r_7 arrives at its next city. This coordination continues through the remaining frames. Subfigures *(c)* and *(d)* show r_1 attempting active coordination with r_0 and r_0 being unable to assist; simultaneously, r_1 plans a better alternative path (the thick dashed path) that enables it to maintain constraints using passive coordination. These subfigures also show r_{11} and r_{12} actively coordinating: r_{12} pays r_{11} to take a longer path around the occluding obstacle. To maintain its own constraints, r_{11} will shortly begin using r_1 as its uplink (not shown). These four frames span thirty seconds of a three minute mission.

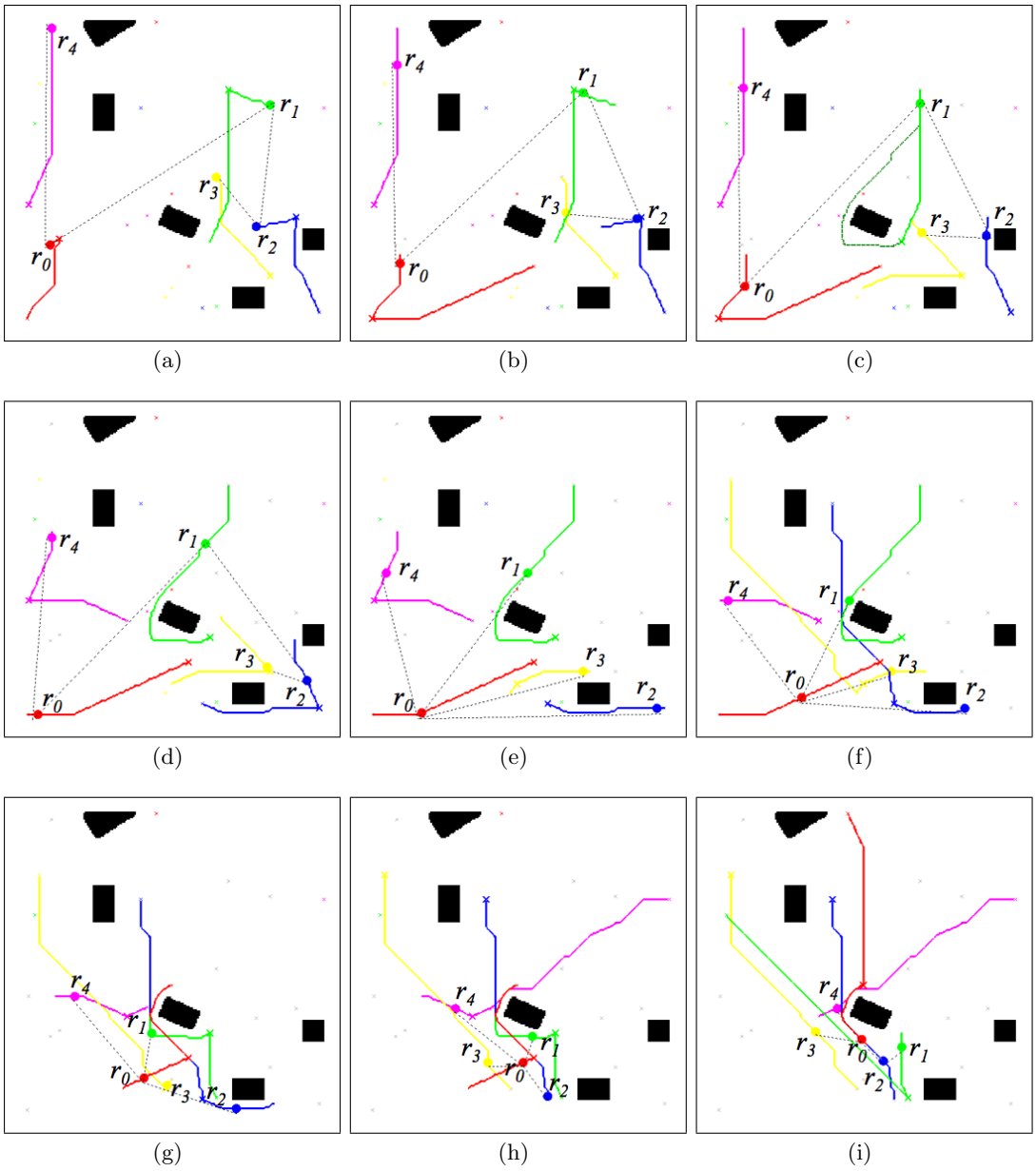


Figure 7.21: These nine snapshots are the first half of a simulation experiment with five robots and twenty cities. Snapshots are taken at five second intervals. Frame *(a)* shows the initial problem setup. In Frame *(c)*, r_1 considers two competing paths: one is quickest to its next goal while the other maintains connectivity with r_0. In Frame *(e)* we see every robot using r_0 as its uplink; the team's tree structure has evolved from a depth of four to a depth of one.

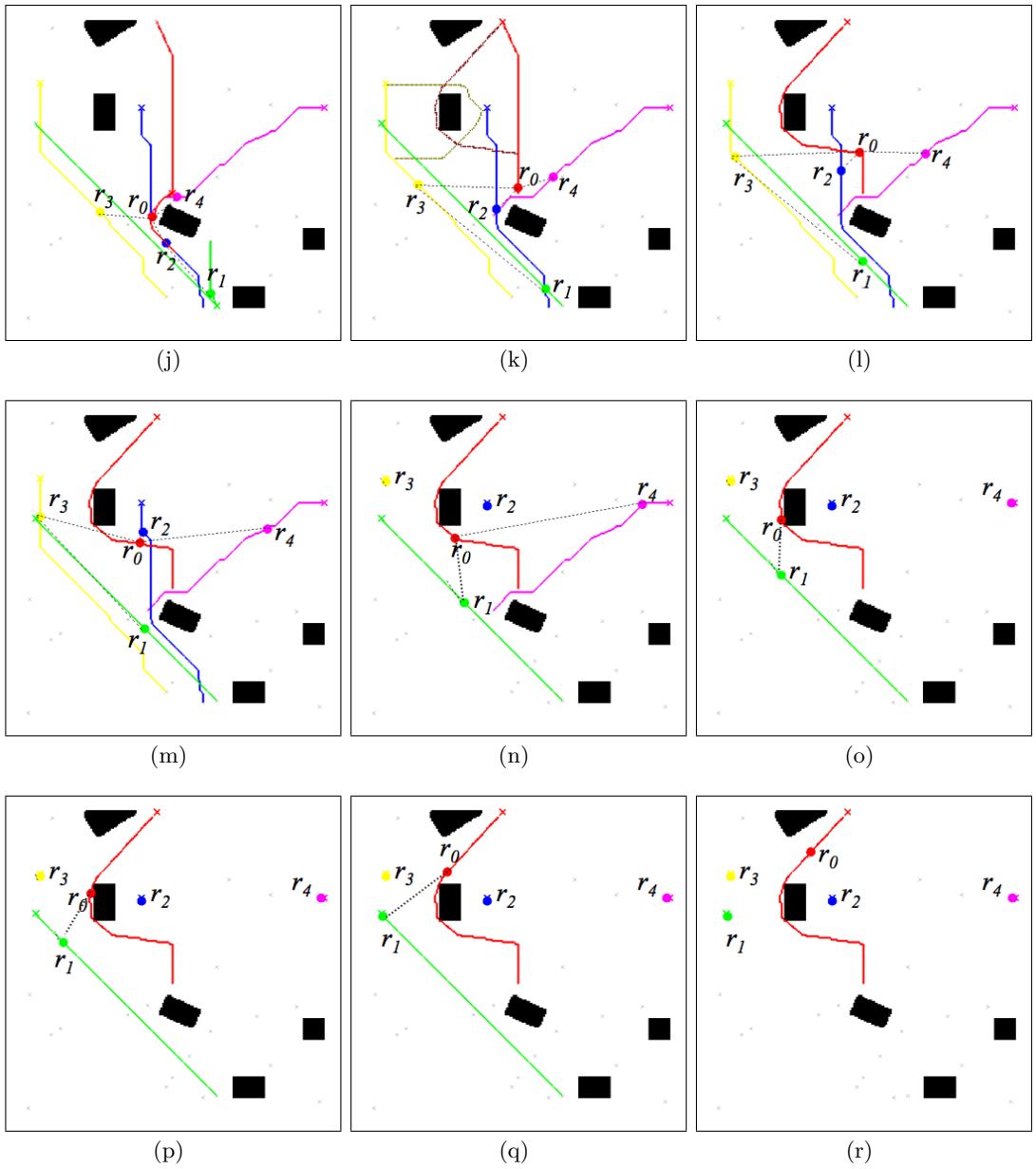


Figure 7.22: These nine snapshots are the second half of a simulation experiment with five robots and twenty cities. In Frame *(k)*, r_3 tries to actively coordinate with r_0 to maintain connectivity; the paths each one is considering during the negotiation are also shown. From Frame *(l)* we see that r_3 has purchased r_0's participation in the longer path. Frame *(n)* shows that r_1 also benefits from this negotiation and uses r_0 as its new uplink. As each robot completes its tour, it no longer needs to maintain communication contact.

Chapter 8

Conclusions

This thesis explores the challenges of one of the most difficult classes of real-world tasks for multirobot teams: those that require long-term planning of tightly-coordinated actions between teammates. These tasks involve solving a distributed multi-agent planning problem in which the actions of robots are tightly coupled. Moreover, because of uncertainty in the environment and the team, robots must frequently replan and closely coordinate with each other throughout execution.

8.1 Summary

In Chapter 1, we placed these tasks alongside other multirobot coordination problems and described them more formally by arranging them in a coordinate system with three axes: the degree of coordination required, the degree of planning required, and local manageability. Specifically, our class of tasks requires a very high degree of coordination and long term planning; in the general case, these features describe problems that must be solved centrally and are therefore intractable. Nevertheless, some of these tasks can also be described by a set of locally managed constraints, which means that they can potentially be solved with distributed approaches. This thesis focuses on such tasks which include security sweep and constrained exploration.

As we discuss in Chapter 2, this class of tasks has not been solved effectively by the research community. In general, we find that current approaches that enable tight-coordination use hill-climbing planning techniques to achieve tractability. Conversely, approaches that employ long-term planning to complete the task are limited to those whose components can be decoupled. A few centralized approaches attempt to achieve both, but these are not feasible in the real world because they rarely scale, tend to be unresponsive to new information, and suffer from single points of failure.

We have developed a coordination framework called Hoplites in response to the need for

effective approaches to these problems. Although planning for tightly-coupled multirobot systems is a difficult problem, Hoplites solves this problem efficiently by using distributed decision-making whenever possible and centralized planning as required. As we described in Chapter 4, Hoplites is a market-based system that consists of passive coordination and active coordination mechanisms which are tailored to easier and harder problem scenarios, respectively. Passive coordination is light on computation and communication and allows teammates to iteratively respond to each other's actions without directly influencing them. When passive coordination traps robots in local minima, active coordination improves solutions by enabling robots to influence each other directly by buying each other's participation in complex plans over the market. Because Hoplites selectively injects pockets of complex coordination into the system, it provides these improvements while remaining computationally competitive with other distributed approaches. Moreover, this selective complexity allows Hoplites to outperform centralized approaches as well because it can often exploit planners with performance guarantees which a centralized approach cannot. Finally, such benefits are never free; the cost for Hoplites's improvements comes in the form of added implementation complexity, and we discussed the associated design considerations in detail in Chapter 4.

Recall from Chapter 4 that Hoplites is a coordination mechanism and not a planning algorithm. That is, Hoplites decouples a robot's choice of planning algorithms from its choice of coordination mechanism, and it can utilize any planning algorithm or even a set of several planning algorithms as desired. This offers significantly greater flexibility because it allows robots to choose planners according to the difficulty of the problem scenario rather than the complexity of the coordination mechanism. Moreover, when a simple planner fails to find a solution, they can use more complex planners and thus attack problems with increasingly sophisticated algorithms. Most importantly, Hoplites outperforms other approaches because it uses the market to more effectively combine the work of local planners into the global solution. Nevertheless, selecting appropriate planning algorithms is an important step to the team's success since these planners determine the quality of local solutions. Our experience with Hoplites has given us insight into the type of planning algorithms that work well for solving tightly-coupled problems and how they can be incorporated into the framework. We discuss these algorithms and make recommendations in Chapter 5.

Finally, in Chapters 6 and 7, we described in detail our implementation of Hoplites for the security sweep and constrained exploration domains. The different characteristics of these domains allow us to implement and explore the range of features we described in Chapter 4, including different objective and profit functions, team structures, planning methods, and coordination strategies. We conducted a wide range of simulation experiments comparing Hoplites to both distributed and centralized approaches along a number of metrics including solution quality, computation time, and scalability. We also evaluated

its sensitivity to imperfect information and to input parameters and explored its ability to exploit a large planning toolbox. Our results strongly confirm our hypotheses that Hoplites significantly improves solutions by enabling complex coordination over the market and that, by doing so selectively, it remains computationally competitive with other distributed approaches and outperforms even centralized approaches.

8.2 Contributions

This dissertation makes a number of important contributions to the robotics literature.

A general and adaptive approach to addressing our problem space. Hoplites is a market-based coordination framework in which the complexity and strength of the coordination adapt to the difficulty of the problem. In particular, Hoplites uses inexpensive distributed decision-making whenever possible and injects resource-consuming pockets of coupled planning only when necessary. As the experiments show, this adaptiveness enables Hoplites to produce better solutions than both distributed and centralized approaches while remaining computationally competitive with distributed approaches. Additionally, Hoplites is widely applicable to real-world problems because it is general, computationally feasible, scalable, operates under uncertainty, and improves solutions with new information.

First formalization of problem space. Domains such as constrained exploration and security sweep have to date been treated as unrelated though they share many properties. This dissertation is the first to identify their common properties (execution-time coordination that cannot be resolved during task allocation *and* that cannot be planned using hill-climbing), demonstrate these properties empirically, and formalize the resulting problem space in a coordinate system alongside other multi-robot problems. This dissertation also shows that these domains can be solved by distributed approaches if the global constraints are locally managable. In doing so, it clarifies problem requirements and allows us to share solutions between domains that may have previously appeared disparate.

First application of market-based approaches to tight coordination. Until now, market-based frameworks have only been used to decompose and allocate tasks and roles. Indeed, this is also true of intentional approaches to coordination. By enabling robots to buy and sell action-level plans instead of just tasks, Hoplites extends the capabilities and application range of market-based approaches in particular and intentional approaches to coordination in general. In doing so, it paves the way for other work to apply these approaches to tight coordination.

First evaluation of and recommendations for planning algorithms for tight coordination. To date, there are no structured principles for selecting planning algorithms for our problem space. This dissertation outlines a set of general approaches that solve different problem scenarios; these techniques range from simple to complex and span the spectrum of ways in which one can negotiate the tradeoff between speed and completeness. Moreover, it provides guidelines on selecting a set of these techniques that will maximize the likelihood of finding solutions while minimizing redundant computation.

Improvement of previous state-of-the-art coordination framework. This dissertation improves the previous state-of-the-art coordination framework, MVERT [14], by incorporating long-term planning and enabling the communication of intentions. Experiments show that these additions result in a five-fold improvement over the original framework.

8.3 Future work

Our work in this thesis enables us to identify a number of areas of promising future work. These areas include remaining challenges in tight multirobot coordination in general and new opportunities arising from the ideas developed in this thesis in particular. Where applicable, we also discuss limitations of Hoplites.

Complexity

The advantages Hoplites provides come at the expense of greater implementation complexity compared to competing approaches: robots must be able to consider incoming requests for active coordination, track current and future commitments, monitor commitments from other teammates, and pursue mission goals. This complexity limits Hoplites's usefulness if it deters potential users from applying it to their domains even if it provides better solutions than alternative approaches. To make Hoplites easier to use, we would like to develop a Hoplites API that abstracts many standard components such as proposing active coordination, bidding, and tracking contracts and that only requires users to implement domain-specific components such as planning algorithms and local utility functions.

Dynamic Environments

Planning is especially challenging in dynamic environments when it involves tight coordination and constraints between multiple agents. Most approaches to tight coordination in dynamic environments use very short-term planning to maintain responsiveness. When

planning is essential (as in our problem space), however, algorithms that efficiently repair solutions between a set of constrained robots would be valuable; they would be particularly useful additions to Hoplites's planning toolbox. These could be modeled on existing repair algorithms such as D* [107, 108] and Dynamic RRTs[98] which are extremely effective but are not tailored to constrained teams.

Finally, though Hoplites can be operational, it will not be effective in domains where changes occur faster than teammates can form or re-negotiate commitments (e.g., robot soccer). This is because plans will become invalid faster than robots can execute them so the effort in active coordination will be wasted and may in practice slow the system down. Indeed, this is true of any approach that expends significant resources to make long-term decisions.

Robustness

Robustness is a largely unsolved problem for tightly-coordinated teams. Often, tight coordination simply does not allow robustness because teammate failure implies mission failure, as in the case of two robots moving a piano. Other times, it is a domain specific problem. For example, if in constrained exploration a robot malfunctions and leaves its teammate without connectivity, what should its teammate do? We might want the teammate to return to its last connected state, randomly search for other members on the team, or simply continue with its goals until connectivity is restored. Specific opportunities for research include anticipating failures and acting pre-emptively, and recovering from failure where possible by finding new teammates to fill in. This has been achieved for loosely coordinated teams driven by market-based approaches [109, 72], and these techniques may offer insight for our problem space.

Information Sharing

Tight coordination requires significant amounts of information sharing between teammates because accurate information is necessary in order to choose actions that meet the team's goals. In some situations, these requirements can exceed communication capabilities. Approaches are needed that selectively share information between teammates based on some measure of the importance or impact of the information. Some work is currently being done [110] but significant work remains.

Solving general multirobot coordination

Lastly, virtually all existing multirobot approaches solve only one or two components of the coordination problem: task decomposition, task allocation, task scheduling, or coordinated

task execution. While this decoupling makes it easier to focus on specific challenges, in real world problems, these challenges are simultaneously present and often closely coupled (as in constrained exploration). Researchers have come a long way in solving individual parts of the coordination problem, but work to harness these different solutions into complete systems [64] has only just begun. Indeed, the next key research frontier for multirobot coordination is to develop comprehensive systems that address all parts of the problem and that move from academic experiments to implementations in the real world.

Bibliography

[1] D. M. Carroll, C. Nguyen, H. R. Everett, and B. Frederick, "Development and testing for physical security robots," in *Unmanned Ground Vehicle Technology VII: Proceedings of the SPIE*, G. R. Gerhart, C. M. Shoemaker, and D. W. Gage, Eds., vol. 5804, May 2005, pp. 550–559.

[2] J. Brookshire, S. Singh, and R. Simmons, "Preliminary results in sliding autonomy for assembly by coordinated teams," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[3] J. Melton, "Hazardous materials robot," in *Unmanned Systems Capabilities Conference II*, 2005.

[4] P. S. Schenker, T. L. Huntsberger, P. Pirjanian, E. T. Baumgartner, and E. Tunstel, "Planetary rover developments supporting mars exploration, sample return and future human-robotic colonization," *Autonomous Robots*, vol. 14, no. 2-3, pp. 103–126, 2003.

[5] T. Huntsberger, P. Pirjanian, A. Trebi-Ollennu, H. D. Nayar, H. Aghazarian, A. J. Ganino, and M. Garrett, "Campout: a control architecture for tightly coupled coordination of multirobot systems for planetary surface exploration," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 33, no. 5, 2003.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] L. Parker, "An experiment in mobile robotic cooperation," in *Proceedings of Robotics for Challenging Environments*, November 1993.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] H. Osumi, M. Terasawa, and H. Nojiri, "Cooperative control of multiple mobile manipulators on uneven ground," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 1998.

[19] J. Ota, N. Miyata, T. Arai, E. Yoshida, and D. K. J. Sasaki, "Transferring and regrasping a large object by cooperation of multiple mobile robots," in *Proceedings of*

the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 1995.

[20] K. Kosuge, Y. Hirata, H. Asama, H. Kaetsu, and K. Kawabata, "Motion control of multiple autonomous mobile robots handling a large object in coordination," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 1999.

[21] M. Hara, M. Fukuda, H. Nishibayashi, Y. Aiyama, J. Ota, and T. Arai, "Motion control of cooperative transportation system by quadruped robots based on vibration model in walking," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1996.

[22] Z. Wang, E. Nakano, and T. Matsukawa, "Realizing cooperative object manipulation using multiple behaviour-based robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1996.

[23] J. S. Jennings, G. Whelan, and W. Evans, "Cooperative search and rescue with a team of mobile robots," in *Proceedings of the International Conference on Advanced Robotics (ICAR)*, 1997.

[24] F.-C. Lin and J. Y. jen Hsu, "Cooperation and deadlock-handling for an object-sorting task in a multi-agent robotic system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1995.

[25] M. Bowling, B. Browning, and M. Veloso, "Plays as effective multiagent plans enabling opponent-adaptive play selection," in *Proceedings of the International Conference on Automated Planning and Scheduling*, 2004.

[26] D. Vail and M. Veloso, "Dynamic multi-robot coordination," in *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2003 International Workshop on Multi-Robot Systems*. Kluwer Academic Publishers, 2003, vol. 2.

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

[28] P. Pirjanian, C. Leger, E. Mum, B. Kennedy, M. Garrett, H. Aghazarian, S. Farritor, and P. Schenker, "Distributed control for a modular, reconfigurable cliff robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[29] N. Kalra, A. Stentz, and D. Ferguson, "Hoplites: A market framework for complex tight coordination in multi-agent teams," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-04-41, August 2004.

[30] B. Gerkey, S. Thrun, , and G. Gordon, "Parallel stochastic hill-climbing with small teams," in *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2005 International Workshop on Multi-Robot Systems.* Kluwer Academic Publishers, 2005.

[31] N. Kalra, D. Ferguson, and A. T. Stentz, "Incremental reconstruction of generalized voronoi diagrams on grids," in *Proceedings of Intelligent Autonomous Systems*, March 2006.

[32] S. 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.

[33] ——, "A multi-robot cooperative task achievement system," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, April 2000.

[34] D. MacKenzie, "Collaborative tasking of tightly constrained multi-robot missions," in *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2003 International Workshop on Multi-Robot Systems.* Kluwer Academic Publishers, 2003, vol. 2.

[35] 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.

[36] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[37] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE Special Issue on Multirobot Coordination*, vol. 94, no. 7, July 2006.

[38] N. Kalra, D. Ferguson, and A. Stentz, "Hoplites: A market-based framework for complex tight coordination in multi-robot teams," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[39] ——, "Constrained Exploration for Studies in Multirobot Coordination," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2006.

[40] J. Esposito and T. Dunbar, "Maintaining wireless connectivity constraints for swarms and the presence of obstacles," in *ICRA*, 2006.

[41] T. Schouwenaars, A. Stubbs, J. Paduano, and E. Feron, "Multivehicle path planning for nonline-of-sight communication," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 269–290, 2006.

[42] 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.

[43] 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.

[44] M. Koes, K. Sycara, and I. Nourbakhsh, "A constraint optimization framework for fractured robot teams," in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, May 2006.

[45] S. Leroy, J. Laumond, and T. Simeon, "Multiple path coordination for mobile robots: a geometric algorithm," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

[46] C. Bererton, G. Gordon, S. Thrun, and P. Khosla, "Auction mechanism design for multi-robot coordination," in *Advances in Neural Information Processing Systems*, 2003.

[47] C. Bererton and G. Gordon, "Multi-robot coordination in the presence of an adversary," in *Advances in Neural Information Processing Systems*, 2004.

[48] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2001.

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

[50] 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.

[51] G. A. S. Pereira, A. K. Das, V. Kumar, and M. F. M. Campos, "Decentralized motion planning for multiple robots subject to sensing and communication constraints," in

Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings from the 2003 International Workshop on Multi-Robot Systems. Kluwer Academic Publishers, 2003, vol. 267–278.

[52] L. Parker, "ALLIANCE: An architecture for fault-tolerant multi-robot cooperation," *IEEE Transactions on Robotics and Automation*, vol. 14, pp. 504–516, April 1998.

[53] 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.

[54] 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.

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

[56] 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.

[57] G. A. Kaminka and Y. Elmaliach, "Single operator, multiple robots: Call-request handling in tight-coordination tasks," in *Proceedings of Distributed Autonomous Robotic Systems 8.* Springer-Verlag, 2006.

[58] K. Azarm and G. Schmidt, "A decentralized approach for the conflict free motion of multiple mobile robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1996.

[59] M. B. Dias, "Traderbots: A new paradigm for robust and efficient multirobot coordination in dynamic environments," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 2004.

[60] 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.

[61] R. Emery-Montemerlo, "Game-theoretic control for robot teams," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, August 2005.

[62] L. Chaimowicz, V. Kumar, and M. Campos, "A paradigm for dynamic coordination of multiple robots," *Autonomous Robots*, vol. 17, no. 1, 2004.

[63] 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.

[64] E. G. Jones, B. Browning, M. B. Dias, B. Argall, M. Veloso, and A. Stentz, "Dynamically formed heterogeneous robot teams performing tightly-coupled tasks," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2006, submitted.

[65] C. Tovey, M. G. Lagoudakis, S. Jain, and S. Koenig, "The generation of bidding rules for auction-based robot coordination," in *Proceedings of the 3rd International Multi-Robot Systems Workshop, Naval Research Laboratory*, 2005.

[66] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[67] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2002.

[68] R. Zlot and A. Stentz, "Multirobot coordination for complex tasks," *International Journal of Robotics Research Special Issue on the 5th International Conference on Field and Service Robotics*, vol. 25, 2006, to appear.

[69] H. Köse, U. Tatlidede, Çetin Meriçli, K. Kaplan, and H. L. Akin, "Q-learning based market-driven multi-agent collaboration in robot soccer," in *The Turkish Symposium on Artificial Intelligence and Neural Networks*, 2004.

[70] M. B. Dias, B. Browning, M. M. Veloso, and A. Stentz, "Dynamic heterogenous robot teams engaged in adversarial tasks," Robotics Institute, Carnegie Mellon University, Tech. Rep. CMU-RI-TR-05-14, 2005.

[71] M. Golfarelli, D. Maio, and S. Rizzi, "A task-swap negotiation protocol based on the contract net paradigm," CSITE (Research Center for Informatics and Telecommunication Systems), University of Bologna, Tech. Rep. 005-97, 1997.

[72] M. B. Dias, M. Zinck, R. Zlot, , and A. Stentz, "Robust multirobot coordination in dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[73] M. B. Dias and A. Stentz, "Opportunistic optimization for market-based multirobot control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[74] N. Nilsson, *Principles of Artificial Intelligence*. Tioga Publishing Company, 1980.

[75] A. Stentz, "Optimal and efficient path planning for partially-known environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 1994.

[76] 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.

[77] 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.

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

[79] J. P. van den Berg and M. H. Overmars, "Prioritized motion planning for multiple robots," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2005.

[80] S. LaValle and J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[81] J. E. Borrett, E. P. K. Tsang, and N. R. Walsh, "Adaptive constraint satisfaction: the quickest first principle," in *12th European Conference on Artificial Intelligence*, 1996.

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

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

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

[85] 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.

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

[87] 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.

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

[89] 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.

[90] 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.

[91] 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.

[92] 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.

[93] 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.

[94] H. G. Tanner, G. J. Pappas, and V. Kumar, "Leader-to-formation stability," in *ICRA*, 2003.

[95] 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.

[96] J. O. Cerdeira, "The multi-depot traveling salesman problem," *Investigação Operacional*, vol. 12, no. 2, 1992.

[97] G. Laporte, Y. Nobert, and H. Mercure, "The multi-depot travelling salesman problem," *Methods of Operations Research*, vol. 40, 1981.

[98] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with RRTs," in *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2006.

[99] D. Ferguson and A. Stentz, "Anytime RRTs," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[100] J. Vazquez and C. Malcolm, "Distributed multirobot exploration maintaining a mobile network," in *International IEEE Conference on Intelligent Systems*, 2004.

[101] H. G. Nguyen, N. Pezeshkian, A. Gupta, and N. Farrington, "Maintaining communication link for a robot operating in a hazardous environment," in *Proceedings of the ANS International Conference on Robotics and Remote Systems for Hazardous Environments*, March 2004.

[102] S. Cabello, Y. Liu, A. Mantler, and J. Snoeyink, "Testing homotopy for paths in the plane," *Discrete and Computational Geometry*, vol. 31, no. 1, January 2004.

[103] M. Powers and T. Balch, "Value-based communication preservation for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.

[104] M. G. Lagoudakis, M. Berhault, S. Koenig, P. Keskinocak, and A. J. Kleywegt, "Simple auctions with performance guarantees for multi-robot task allocation," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[105] M. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing," in *Robotics: Science and Systems*, 2005.

[106] J. Beardwood, J. H. Halton, and J. M. Hammersley, "The shortest path through many points," *Proceedings of the Cambridge Philosophical Society*, vol. 55, no. 299–327, 1959.

[107] A. Stentz, "The focussed D* algorithm for real-time replanning," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[108] S. Koenig and M. Likhachev, "D* lite," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2002.

[109] B. P. Gerkey and M. J. Matarić, "Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2002.

[110] M. Roth, R. Simmons, and M. Veloso, "What to communicate? execution-time decision in multi-agent pomdps," in *Distributed Autonomous Robotic Systems*, 2006.