Search in the Physical World

Geoffrey A. Hollinger

CMU-RI-TR-10-20

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy in Robotics

Robotics Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

July, 2010

Thesis Committee:

Sanjiv Singh (chair)

Geoffrey Gordon

Reid Simmons

Athanasios Kehagias, Aristotle University of Thessaloniki

Abstract

This thesis examines search in the physical world, which differs significantly from the searches in the digital world that we perform every day on our computers. When searching the internet, for instance, success is a matter of informed indexing that allows the information to be retrieved quickly. In these cases, there is no consideration of the physical nature of the world, and the search is not cognizant of space, time, or traversal distance. In contrast, search in the physical world must consider a target that could be continuously moving, possibly even trying to evade being found. The environment may be partially known, and the search proceeds with information gathered during the search itself. In many cases, such as guaranteeing capture of an adversarial target, the problem cannot be solved with a single searcher, and all group members must coordinate their actions with others on the team. Prior work has explored limited instances of such problems, but existing techniques either scale poorly or do not have performance guarantees.

Two of the main variations of search in the physical world are considered: efficient search and guaranteed search. During efficient search, robots move to optimize the average-case performance of the search given a model of the target's motion. During guaranteed search, robots coordinate to minimize the worst-case search time if the target is adversarial. This thesis unifies these search problems and shows them to be NP-hard, which suggests that a scalable and optimal algorithm is unlikely. In addition, it is shown that efficient search admits a bounded approximation, and guaranteed search does not. Despite these hardness results, algorithms using implicit coordination can provide scalable and high-performing solutions to many real-world search problems. Implicit coordination is defined as the sharing of locations, measurements, and/or actions to improve the team plan. In accord with this design strategy, a linearly scalable efficient search algorithm is presented that utilizes implicit coordination to achieve bounded performance. In addition, this thesis contributes a novel approach that augments the coordination with a pre-search spanning tree generation step, which leads to an anytime algorithm for guaranteed search.

With a focus on decentralized and online operation, the proposed search algorithms are extended to take into account team constraints, limited communication, and partially known environments. The techniques are illustrated using a scenario from the literature that incorporates both efficient and guaranteed search, and they are validated both in simulation and on human-robot search teams operating in the physical world. The developed framework enables teams of autonomous agents to search environments outside the scope of previous techniques, and the analysis provides insight into the complexity of multi-robot coordination problems.

Acknowledgments

First and foremost, I would like to acknowledge the guidance, support, and patience of my advisor Sanjiv Singh. I know Sanjiv and I will have many more productive discussions (and arguments) in the years to come. I am also grateful for the revisions and helpful suggestions provided by my committee members, Athanasios Kehagias, Geoffrey Gordon, and Reid Simmons. Further thanks go to Joseph Djugash, Benjamin Grocholsky, Siddhartha Srinivasa, Dave Ferguson, Matt Aasted, Seth Koterba, and Christopher Geyer for their insightful comments. I thank Debadeepta Dey, Pragun Goyal, and Bradley Hamner for assistance with experiments, and Sanae Minick and Suzanne Lyons Muth for help with all the gory details of graduate school.

The SDR and Merced maps were obtained from the Robotics Data Set Repository (RADISH) (Howard and Roy, 2003); thanks to Andrew Howard for providing this data. Multispectral Solutions Inc. was kind enough to loan ranging radios for some of the experiments in this thesis. Portions of this work were funded by the National Science Foundation under Grant No. IIS-0426945, and portions were funded by Intel Research Pittsburgh under a summer student fellowship. Further thanks go to the Boeing Company and the Robotics Institute for financial support.

The friends who have helped me along the way are too numerous to mention, but every one of them deserves a hearty thank you. My family members are somewhat less numerous, so I will go ahead and mention them. Thanks to Sarah for supporting me (and putting up with me) through this process, and thanks to Mike and Cathy for providing great holiday escapes. Further thanks to Gram and George for always having confidence in my abilities. A warm thank you to my parents, without whom I could not have written this dissertation. Finally, a shout-out to Zach, the next Hollinger to follow his dreams.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

If Edison had a needle to find in a haystack, he would proceed at once with the diligence of the bee to examine straw after straw until he found the object of his search. . . . I was a sorry witness of such doings, knowing that a little theory and calculation would have saved him ninety per cent of his labor.
— Nikola Tesla, New York Times, October 19, 1931

1.1 The Search Problem

IMAGINE you are with a group in a large, complex building like a museum, supermarket, or office. You suddenly realize that a member of the group is missing. You now need to coordinate the group to find the lost person. A few considerations will likely cross your mind. First and foremost, you need to determine the group's search strategy. One option is to take a centralized approach, which sets one person as the commander to formulate the search strategy for the entire group. However, the commander would need to consider a large number of potential strategies to plan the actions of the entire team, and the execution of the chosen strategy may not even be possible with limited communication. An alternative decentralized option is for each group member to determine his or her own path without the guidance of a commander. Group members can improve their search paths by sharing information about where they are and where they are going. Numerous variations also exist, including assigning parts of the building to each group member and dividing into teams.

You may be in a hurry, and you would like to minimize the time it takes to find the person. In other words, you need to guide the search to find the target efficiently. This variation will be referred to as *efficient search*: the problem of locating a target in the minimum expected time. Efficient search allows you to utilize information about how the person moves. For example, you know that the lost person is walking and can only move with bounded speed. You may also have some knowledge of particular parts of the building in which the person is likely to have been. This knowledge would lead you and the group to search the more likely places first (e.g., the hallways and large spaces).

Desirable efficient search paths are those that on average are likely locate the target quickly. If noisy measurements of a target's position are available, the problem is one of both estimation and control. An example would be if someone told you the lost group member was seen in a neighboring room but was unsure which one. If this information is properly utilized, better estimation can lead to better search paths. However, perfect estimation is useless without good control strategies.

After searching fruitlessly for a while, you may wonder if its possible to coordinate the group in such a way that you are guaranteed to find the lost group member. A second variation of coordinated search is *guaranteed search*, where searchers work together to scour the environment to ensure that they find a target if one exists. Taking a guaranteed search approach would allow you to find the lost group member even if she were acting differently from your expectations (e.g., standing still or moving extremely quickly). You also note that, depending on the complexity of the environment, the number of searchers may be insufficient to guarantee finding the lost group member.

Depending on the environment, there may be important constraints on your search. For instance, you may want to maintain a line-of-sight chain to avoid losing another group member. Similarly, if you are using walkie-talkies, you may want to minimize the distance between team members to ensure communication. The *constrained search* problem requires either efficient or guaranteed search while maintaining some constraints on the searchers' movement. These constraints makes the problem more difficult by restricting the number of paths that the searchers can traverse. This problem also necessitates tight coordination between the searchers to satisfy the constraints.

In addition, you may only have a partial map, and you would like to utilize new information about the building while you are searching. For instance, your current map of a museum may not include a new wing. If the lost group member has wandered into this wing, the group will need to modify the search plan to take into account new information about that area.

Prior research in searching often takes place on abstract graphs or simple environments. Rarely are the algorithms scalable to multiple searchers, and they often require a centralized planner to coordinate the entire team. Furthermore, the searchers often are assumed to have perfect communication and perfect information about the environment. In the cases where rigorous analysis is employed, the assumptions are often overly restrictive and preclude operation in real-world environments.

Internet search engines like Google also solve a variant of the search problem in a digital world. However, searching the internet is very different from searching the physical world. Search in the digital world optimizes retrieval from an extremely large amount of information that is relatively static. Locality and distance constraints are related to cache structure, in which some pairs of requests are faster when made together. In contrast, search in the physical world must consider a target that could be continuously moving, possibly even trying to evade being found. The environment may be only partially known ahead of time, and the search proceeds with information gathered during the search itself. Each group member also faces the problem of coordinating their search with other searchers whose actions are distinct from one another.

1.2 Thesis Statement

This thesis examines the problem of searching the physical world with multiple autonomous robots and develops high-performing solutions to the efficient, guaranteed, and constrained search problems. In general, finding the optimal solution to multi-robot search problems requires computation exponential in the number of searchers, which becomes computationally infeasible for large teams and large environments. The complexity arises because multiple searchers must tightly coordinate to achieve the optimal solution. Tight coordination requires considering the joint action space for all searchers, which grows exponentially in the number of searchers. Considering the joint action space is an example of *explicit coordination* during which the searchers explicitly plan for their teammates. Solving the coordinated search problem using a centralized Partially Observable Markov Decision Process (POMDP) is an example of an explicitly coordinated solution (Kaelbling et al., 1998). Solving POMDPs optimally, or even near-optimally, is a notoriously difficult problem, which motivates the development of scalable approximate algorithms for multi-robot search problems.

Alternatively, if each searcher plans individually without taking into account the future actions of its teammates, the size of the search space does not increase. Since the searchers are no longer coordinating in any way, this is an instance of *no coordination*. Paths generated without any coordination often perform poorly because the searchers have no mechanism for reasoning about their teammates' actions. Market-based techniques use synthetic auctions to solve tightly coordinated tasks by explicitly coordinating only when necessary (Gerkey and Mataric, 2002; Kalra, 2006). Such an approach is one way to strike a balance between explicit coordination and no coordination at all, but determining when to coordinate explicitly and how much explicit coordination is necessary is a difficult problem in itself.

If searchers share their paths, they provide valuable information that can improve the joint search plan. In this case, the searchers are not explicitly planning for their teammates, but they are implicitly coordinating by sharing information. Examples of shared information include the searchers' positions, planned paths, and estimates of the target's position. The algorithms in this thesis utilize *implicit coordination* to achieve better scalability than centralized approaches. An implicitly coordinated solution is one in which robots share information but do not plan the actions of their teammates. For some loosely coupled multi-robot problems, implicit coordination is sufficient to provide near-optimal solutions. This thesis shows that efficient search falls into this category of problems.

This thesis also applies an implicit coordination algorithm to the constrained search problem, where searchers must maintain connectivity constraints during search (e.g., line-of-sight or range connectivity). The idea of periodic connectivity is introduced, which requires searchers to regain connectivity once every fixed interval. Periodic connectivity relaxes the typical assumption made in prior work that multi-agent teams must maintain connectivity at all times. This thesis shows that relaxing the connectivity constraints using periodic connectivity allows for the application of implicit coordination in this tightly coordination domain.

The use of implicit coordination can lead to poor solutions in some domains that require tight coordination. This is a particularly prominent concern when considering an adversarial target, since the searchers must work together to make any progress. To improve performance for these problems, searchers can execute a shared preprocessing step, which transforms the environment representation into one solvable through implicit coordination. This thesis shows how utilizing spanning trees of the environment can yield implicitly coordinated search strategies, leading to an "anytime" algorithm for guaranteed search. An anytime algorithm is proposed that finds a feasible solution and then improves this solution over time. The proposed approach yields feasible guaranteed search paths even in very large environments.

Algorithms using implicit coordination have the advantage of being *decentralized*. In the context of this thesis, decentralized will be defined by the following qualities: a) operates without a central controller and b) does not have a single-point failure. In addition, the algorithms in this thesis are capable of running *online* during the search, either through constant replanning or replanning after failure. Online operation allows for the incorporation of new information as it becomes available, which is particularly desirable when operating in the physical world.

The theoretical and experimental results in this thesis demonstrate that implicit coordination combined with an informed environment representation can generate search strategies competitive with those produced by explicit coordination while using far fewer computational resources. From here follows the thesis statement:

> *Algorithms using implicit coordination provide scalable, decentralized, online, and high-performing solutions to multi-robot search problems in complex physical environments.*

The problem of coordinating teams of mobile robots to search large environments in the physical world is relevant to many important applications. Military and first response teams often need to locate lost team members or survivors in disaster scenarios. The increasing use of search and rescue robots and mechanized infantry necessitates the development of algorithms for autonomously searching such environments. In addition, autonomous naval search applications are becoming more prominent as unmanned surface and underwater vehicles increase in capabilities. Autonomous surface and underwater vehicles have the potential to improve harbor surveillance and open ocean search.

Another application of multi-robot search is locating a lost person in an indoor environment. In one such scenario, a moving first responder is lost during disaster response, and a team of robots must locate him or her. Noisy measurements of the first responder's position may or may not be available to assist in the search. A similar scenario arises if a group of ground vehicles must locate a target on a road network while an air vehicle provides surveillance. Finally, other relevant application exist in quality of life technologies for offices, personal residences, and nursing homes. Locating objects to assist daily living and even finding patients in care facilities has the potential to improve quality of life technologies. Together, these applications provide strong motivation for the study of autonomous search.

1.3 Formulation

1.3.1 Environment

In this thesis, it is assumed that K searchers must locate one or more targets (or adversaries) in some finite-sized environment $\mathcal{E}$. The environment is discretized (see Chapter 3) into an undirected graph $G = (N, E)$, where the nodes N represent convex regions in the environment, and the edges E represent connections between these regions. For most of this thesis, the environment graph is assumed to be known and unchanging. A 2D environment would necessarily yield a planar graph, whereas a higher dimensional environment may not. The examples in this thesis utilize 2D environments, and hence planar graphs, but the proposed algorithms are applicable to general graphs and higher dimensional environments.

1.3.2 Searchers

At a given time t, one or more searchers exist on each node $u \in N$, and they may move to a node $v \in N$ at time $t+1$ if there exists an edge $uv \in E$. The location of searcher k at time t is denoted by $s_k(t)$. A searcher's path for times 1 to some end time T is denoted as $A_k = s_k(1) \ldots s_k(T)$, and $A_k(t)$ for a given time t. The combined paths for all searchers is denoted as A or $A(t)$ for a given time t.

Definition All possible paths that the searchers may take on the graph are referred to as the joint path space. Consider the possible searcher locations $s_k(t) \in N$ of K searchers at times $t \in \{1, 2, \ldots, T\}$. The searchers' configuration space at time t is $\Phi(t) = \{s_1(t), \ldots, s_K(t) | s_1(t) \in N, \ldots, s_K(t) \in N\}$. The searchers' joint path space is defined as the Cartesian product $\Psi^S = \Phi(1) \times \ldots \times \Phi(T)$.

1.3.3 Target

A target also exists on the nodes of the graph at locations $e(t)$. This thesis deals primarily with a single target, but extensions to multiple targets are often straightforward. The target's path will be referred to as Y and its path space as Ψ^E. If the target's movement is independent of the searchers' locations and plans, the target is considered non-adversarial. If the target utilizes the locations or plans of the searchers to avoid capture, it is considered adversarial (and may be referred to as an evader). In some cases, the target may also be capable of moving more than one node in a single time step. An adversarial target with perfect knowledge of the searchers' plans and infinite speed is considered a worst-case target.

A target is found (or captured) if any searcher exists on the same node as the target at the same time, i.e. $s_k(t) = e(t)$, for any k. In some cases, there may be a nonzero probability of a failed capture even if the target is co-located with a searcher (due to sensor noise, for instance). The searchers' goal is to find the target in fewest number of time steps. It is assumed that no target may pass through a node occupied by a searcher without a chance of a capture event occurring.

1.3.4 Search Optimization

For every target path $Y \in \Psi^E$, define a function $F_Y(A)$ of the searcher paths $A \in \Psi^S$. Let $F_Y(A)$ be a function that is inversely proportional to the capture time (i.e., discounted time to capture). If the target's motion is non-adversarial, and a probabilistic model $\Pr(A)$ is known for each path, the searchers' optimization problem is defined below:

$$A^* = \operatorname*{argmax}_{A \in \Psi^S} \sum_{Y \in \Psi^E} \Pr(Y) F_Y((A) \tag{1.1}$$

If the target is adversarial, and it can choose the path the best minimizes the function $F_Y(A)$ given perfect knowledge of the pursuer's paths, then the searchers' optimization problem is defined below:

$$A^* = \operatorname*{argmax}_{A \in \Psi_S} \min_{Y \in \Psi^E} F_Y(A) \tag{1.2}$$

If the searchers' goal is maximize the average-case reward for a non-adversarial target as in Equation 1.1, then the problem will be referred to as efficient search (see Chapter 3). Conversely, if the searchers' goal is to assure capture of a worst-case adversarial target as in Equation 1.2, then the problem will be referred to as guaranteed search (see Chapter 4). In some cases, the searchers may not know whether a target is acting adversarially or non-adversarially. In such a case, both worst-case and average-case performance of the search must be considered (see Chapter 5).

In some applications, the searchers' path space Ψ^S will be further limited by constraints between searchers. For instance, configurations that disconnect some searchers from line-of-sight and/or range contact with the rest of team may be considered invalid. When inter-robot constraints are imposed, determining if A_i is feasible for robot i requires considering A_j for some $j \neq i$. In these cases, the problem will be referred to as constrained search (see Chapter 6). Note that searchers performing constrained search will also plan to optimize either Equation 1.1 or Equation 1.2 depending on the type of target.

1.3.5 Implicit Coordination

Definition Implicit coordination is when each searcher only considers changing its own path while planning. More precisely, searcher k only examines paths in the space Ψ_k^S while considering its next move, though it may utilize plan and/or state information shared by other searchers.

Implicit coordination arises when searchers share their intentions to improve their own plans, which may be done synchronously or asynchronously. For instance, searchers may plan in a sequential order with each searcher waiting for the previous searcher to plan and share its path. Alternatively, agents may plan asynchronously when they reach planning points in the environment, and then share their paths at this time.

In general, implicit coordination does not imply any commitment that a plan will be carried out to completion. If one team member determines that it should modify its plan, it should immediately share that information with the rest of the team. If a delay in communication is present, asynchronous implicit coordination can suffer from race conditions, which will negatively affect performance. Avoiding these race conditions is necessary for the successful application of implicit coordination. In the domains in this thesis, race conditions are rare because searchers are usually not planning at the same time. In the cases where simultaneous planning is required, a sequential approach is employed where some searchers have precedence over others.

1.4 Thesis Summary

This thesis utilizes implicit coordination to provide a scalable, unified approach to search with multiple autonomous robots. The proposed algorithms are analyzed theoretically and validated both in simulation and on heterogenous search teams operating in the physical world. The remainder of this thesis is organized as follows.

Chapter 2 describes related work in graph search, pursuit-evasion, probabilistic planning, and sensor networks. A survey of the literature shows that prior work does not take into account all considerations necessary for coordinated search in complex environments. In addition, prior work does not consider a unified approach to average-case and worst-case search as described in this thesis.

Chapter 3 proposes Finite-Horizon Path Enumeration with Sequential Allocation (FHPE+SA), a receding horizon implicit coordination algorithm for solving the efficient search problem. Performance guarantees for implicitly coordinated solutions are shown by utilizing the theoretical property of submodularity. The proposed approach is validated through simulations and experiments, some of which utilize ranging radio sensors capable of receiving range measurements between agents without line-of-sight. A formulation of the efficient search problem as a Partially Observable Markov Decision Process (POMDP) is also shown, and FHPE+SA is compared to a number of general POMDP solvers and heuristics.

Chapter 4 proposes Guaranteed Search with Spanning Trees (GSST), an anytime algorithm for the guaranteed search problem that utilizes search schedules for underlying spanning trees to guide search on general environment representations. The approach is compared to several other guaranteed search algorithms, and it is shown to provide solutions with fewer searchers while remaining computationally feasible for large teams and environments.

Chapter 5 unifies the efficient and guaranteed search problems into a common framework, and provides formal hardness results for the unified problem. A combined search algorithm is proposed that generates search schedules that perform well under both metrics, and experiments are shown using a human-robot search team in an office environment. The proposed approach is extended to consider partially known environments and imperfect motion models.

Chapter 6 introduces the problem of constrained search with periodic connectivity. Implicit coordination is applied to this domain and compared to methods that require continual connectivity. The proposed approach is implemented on mobile robots performing a constrained search task in an outdoor environment. This chapter also provides general theoretical hardness results for connectivity constrained multi-robot path planning problems. In addition, a passive data fusion technique is introduced for searchers that become disconnected for long periods of time.

Chapter 7 explores the problem of combining search and action with application to personal assistance robotics. The problem of searching for an object and then performing a useful action with that object is explored, and theoretical analysis is given showing when it is necessary to consider the cost of the subsequent action to determine the optimal search strategy. This chapter extends the implicit coordination algorithms described previously to include this case. Results are shown on a state-of-the-art mobile manipulator operating in an office environment.

Chapter 8 provides further validation of the algorithms in this thesis using scenarios from the Multi Autonomous Ground-robotic International Challenge (MAGIC) 2010 competition. It is shown that modifications of the proposed efficient and guaranteed search algorithms provide robust solutions to the MAGIC search and secure problem. The proposed algorithms are extended to deal with searcher failures using dynamic replanning. Robustness trials are presented in simulation for various failure modes, which demonstrate the effectiveness of the proposed approach.

Finally, Chapter 9 summarizes the contributions of this thesis and introduces avenues for future work. The analysis throughout this thesis is utilized to highlight the characteristics of multi-robot coordination problems that allow for the successful application of implicit coordination. This chapter also explores broader applications of implicit coordination to a wide range of multi-robot tasks, including exploration, monitoring, and tracking.

Chapter 2

Related Work

THIS thesis draws on a wide breadth of work from graph search, pursuit-evasion, probabilistic planning, and sensor networks. This chapter divides the related work in robotic search into sub-areas and examines the state-of-the-art in each area. The advantages of previous methods are compared with those of the proposed method in this thesis. This section demonstrates the need for a unified framework to handle all the demands of searching in the physical world.

2.1 Pursuit-Evasion

Early research in coordinated search almost exclusively treated the target as an omniscient, adversarial evader that actively avoids capture. Parsons (1976) developed some of the earliest methods for solving the adversarial pursuit-evasion problem on graphs. He considered the graph to be a system of tunnels represented by the edges of the graph in which an evader was hiding, and he defined the *search number* of a graph to be the minimum number of pursuers necessary to catch an adversarial evader with arbitrarily high speed. Determining the search number of a graph was later found to be NP-hard (Megiddo et al., 1988), and to be NP-complete due to the monotonicity of optimal edge search schedules (Bienstock and Seymour, 1991; LaPaugh, 1993). In this early work in pursuit-evasion, the evader can only hide in the edges of the graph (referred to as *edge search*), which does not fit with the intuitive representation of many environments (e.g., the rooms of a building naturally correspond to nodes, not edges of a graph). Early researchers were primarily concerned with examining the hardness of the guaranteed search problem on graphs, and they did not present algorithms for guaranteed search that are scalable to large teams in realistic environments.

Edge search does not apply directly to many robotics problems. The possible paths of an evader in many indoor and outdoor environments in the physical world cannot be accurately represented as the edges in a graph. In some cases, it is possible to construct a dual graph by replacing the nodes with edges, but these translations do not necessarily yield the same results as the original problem (Kehagias et al., 2009a). To better model physical environments, robotics researchers have studied alternative formulations of the guaranteed search problem. Guibas and LaValle et al. extended pursuit-evasion techniques to guarantee capture in polygonal environments (LaValle et al., 1997; Guibas et al., 1999). Their approach discretizes polygonal environments into conservative visibility regions and then uses the resulting information space to develop complete algorithms that guarantee capture in one-searchable graphs. Gerkey et al. (2006) applied these ideas to searchers with limited sensing. For a single pursuer, these algorithms are guaranteed to find a solution if one exists. To maintain completeness with multiple searchers, these algorithms would need to generate a potentially exponential number of visibility regions and search the resulting information space. This approach is only tractable for few searchers and small environments. An alternative is to use an iterative visibility-based method (Guibas et al., 1999), but doing so loses completeness and becomes similar to uninformed greedy planning (see Chapter 4).

Other researchers have examined variations of the pursuit-evasion problem. Adler et al. (2003) introduced the *escape length* of a strategy as the worst-case expected number of rounds for the pursuer to catch the evader, and they derived error bounds for various randomized pursuer strategies. Isler et al. (2005) advanced the concept of randomized searcher strategies to polygonal environments. They developed coordination strategies for one or two pursuers in simple polygonal environments based on the assumption that an adversarial evader does not have knowledge of some actions made by the pursuer. These algorithms are designed for small teams of searchers and are not easily extendable to larger problems.

Kolling and Carpin (2010) showed how a polynomial-time graph cut algorithm can be applied to a weighted graph formulation of the multi-robot surveillance domain. Their algorithm operates on an alternative formulation of the clearing problem, which requires several searchers to clear areas in the environment. They consider a probabilistic variant with imperfect capture sensors (Kolling and Carpin, 2009), but they do not provide a mechanism to utilize information about the target's motion beyond a worst-case assumption. In addition, many of their methods are limited to cases where the surveillance graph is a tree.

2.2 Graph Theoretic Search

Graph theoretic search literature discusses several variations of the guaranteed search problem. Fomin and Thilikos (2008) recently provided a comprehensive survey of recent results in guaranteed graph searching (a slightly earlier one was provided by Alspach (2006)), which includes formulations for edge search, node search, evaders with limited speed, omniscient evaders, and cases where the location of the evader is visible to the searchers. Several alternative versions of "node search" appear in the literature. In one formulation, the evader resides in the edges of the graph (hence, despite the name, this is really an edge search problem), and these edges are cleared by trapping (i.e., two searchers occupy the adjacent nodes). In this thesis, we are primarily concerned with the variant of node search in which the both the searchers and targets exist on the nodes of the graph.

Kehagias et al. (2009a) discussed many of the theoretical properties of node search, and they show its formal relationship to edge search. Both edge search and node search are NP-hard, and all search schedules that clear the edges of a graph also clear the nodes of the graph. On the contrary, all schedules that clear the nodes of the graph do not necessarily clear the edges of the graph. Thus, edge search clearing schedules can be utilized as node search clearing schedules but not necessarily vice versa. In addition, a variant called mixed search, in which edges can be cleared by trapping (i.e., an edge is cleared if a searcher is located at both ends), was shown by to be equivalent to edge search. This analysis allows many of the principles and algorithms from edge search to be reused for node search.

The traditional formulation of guaranteed search does not restrict the movement of searchers. In other words, searchers are allowed to "teleport" between nodes in the graph without following the edges between them. This assumption enables searchers to clear disjoint parts of the graph without maintaining a route to a starting node. Barrière et al. (2003) introduced the idea of connected search, during which searchers must maintain a connected subgraph of cleared nodes. Connected search guarantees that a path exists to the starting nodes at all times and that searchers are connected by a cleared or "safe" region of the graph. Barrière et al. argue that such a constraint is an important quality for search strategies in the network decontamination domain.

Connectedness is also an important characteristic of guaranteed search strategies in the physical world. Real robots cannot teleport between nodes in the graph because these nodes represent physical locations. Instead, robots must restrict their search paths to those traversable in the environment. Furthermore, domains like urban search and rescue and military reconnaissance require a safe path back to the starting point to aid in evacuation. These applications motivate the examination of connected search paths during guaranteed search. However, there is a "price of connectedness" because clearing a graph with a connected search schedule may require more searchers than with an unconnected one (Fomin et al., 2004). Also, it is important to note that connected search constrains the cleared set and not the searcher configurations, which differs from enforcing constraints between agents as discussed in Chapter 6.

Early research in guaranteed search showed that many variations of the problem are NP-hard on arbitrary graphs (Megiddo et al., 1988). This result suggests that finding exact solutions to the guaranteed search problem on large graphs is intractable (unless $P = NP$). The formal hardness of search problems has motivated researchers to develop guaranteed search algorithms on special cases of graphs. Barrière et al. (2002) showed that an optimal connected search schedule with minimum searchers can be found in linear time on trees. Polynomial-time algorithms for searching hypercubes

(Flocchini et al., 2008), tori, and chordal rings (Flocchini et al., 2007) have also been developed. Unfortunately, most environments in the physical world cannot be represented by these special cases. The guaranteed search algorithm in this thesis extends the advantages of special case solutions by transforming a graph into its spanning tree through the use of guards and then running an exact search algorithm on the spanning tree (see Chapter 4).

Since many applications require guaranteed search in arbitrary environments, prior work has proposed some approximate algorithms that provide good performance. Flocchini et al. (2005) examined a genetic algorithm approach for clearing arbitrary graphs. Their approach does not take into account prior information about the environment, and it does not allow for extensive coordination between searchers. Thus, it can require many searchers (more than the optimal) in fairly simple environments. Kehagias et al. (2009b) showed that iterative greedy node search (IGNS), a dynamic programming inspired algorithm that iteratively maximizes the number of cleared nodes, can work well on complex graphs. However, this algorithm is unable to find a clearing schedule with the minimal number of searchers on many complex graphs (see Chapter 4). On the other hand, this algorithm can produce nonmonotonic searches and can deal with both finite evader speed and non-local visibility.

A key insight in the development of approximate guaranteed search algorithms is the connection between graph search and the graph parameters of treewidth and pathwidth. A tree decomposition of a graph is a new graph, which (a) is a tree; (b) has nodes which correspond to *sets of nodes* of the original graph (these new "supernodes" are called *bags*); (c) satisfies some additional technical conditions (listed in (Dendris et al., 1994)). The *width* of a tree decomposition is the cardinality of its largest bag minus one. The *treewidth* of a graph G is the minimum of the widths of all tree decompositions of G. A minimum width tree decomposition of G yields an optimal clearing schedule for the *visible* search problem (i.e., when the searchers know the location of the target).[1] Similarly, a path decomposition of a graph G is a tree decomposition where the tree is also a path; the *pathwidth* of G is the minimum of the widths of all path decompositions of G; a minimum width path decomposition provides an optimal solution to the guaranteed *invisible* search problem. Approximation algorithms for treewidth and pathwidth have been proposed (Kloks, 1994), but they are not guaranteed to provide connected or internal path decompositions.

Fraigniaud and Nisse (2006) proposed an algorithm for connected search by finding approximately minimal width tree decompositions. They show how these decompositions can be used to find connected search strategies. Though polynomial-time, their algorithm grows in complexity with both the search number and the size of the graph. In addition, their approximation bound degrades fairly quickly with the size of the graph. They do not implement their algorithm, and they do not provide sufficient information to demonstrate that an implementation would be efficient or feasible.

[1]Note that finding a minimal width tree decomposition of an arbitrary graph G also is an NP-hard problem.

The algorithms described above attempt to minimize the number of searchers necessary to clear a given graph. Researchers have paid far less attention to generating clearing schedules that require minimal time or distance. Borie et al. (2009) discussed algorithms and complexity results for the minimal time and minimal distance clearing problems, but they do not provide a scalable algorithm for large teams or graphs. The G-GSST algorithm in this thesis provides clearing schedules with fewer searchers and lower clearing times than competing methods (see Chapter 5).

Guaranteed search algorithms in the literature do not demonstrate “anytime” capabilities. An anytime algorithm is one that quickly generates an initial solution and then improves solution quality with increasing runtime. Anytime algorithms have been successfully applied to POMDP planning (Smith, 2007), dynamic path planning (Likhachev et al., 2005), and many other domains. These state-of-the-art algorithms allow for high-quality solutions with varying levels of computational resources. Zilberstein (1996) discussed some of the desirable qualities of anytime algorithms in intelligent systems. The proposed guaranteed search algorithm in Chapter 4 (GSST) shows many of these qualities, including monotonicity (the solution only improves over time), recognizable quality (the quality of the solution, i.e. number of searchers, can be determined at runtime), consistency (the algorithm will not spend too much time finding a single solution), and interruptibility. To the best of our knowledge, GSST is the first algorithm to bring the advantages of anytime algorithms to the domain of guaranteed search.

The methods above examine only guaranteed search and related problems. Thus, they do not reason about capture time during search, and they do not account for partial knowledge of the target’s location or its motion model. The algorithms in this thesis unify guaranteed search with a probabilistic formulation to reason about target motion and uncertainty. This formulation takes advantage of a non-adversarial assumption on the target’s motion model and its expected behavior in addition to its worst-case behavior. Algorithms in this thesis utilize implicit coordination to allow for scalability to large, complex environments even with an adversarial target. Thus, the techniques perform well in environments outside the scope of previous methods.

2.3 Probabilistic Planning

Both classical pursuit-evasion approaches and more recent guaranteed search algorithms rely on the worst-case assumption of the target’s behavior, which precludes modeling the target’s motion or incorporating uncertainty into the search plans. An alternative formulation of multi-robot search problems use a probabilistic approach to model the location of the target or the movement of the searchers. Many coordinated search problems can be formulated as a Markov Decision Process (MDP) if the target’s position is known or a Partially Observable Markov Decision Process (POMDP) if it is unknown. These formulations provide fully probabilistic representations of the problem, which can easily reason about uncertainty. Eaton and Zadeh (1962) discussed optimal solutions to the MDP guaranteed search problem. Roy et al. (2005) later showed how belief compression can be used to make the POMDP efficient search problem tractable for a single pursuer. A number of approximate sampling-based algorithms are available for solving general POMDPs (Pineau et al., 2003; Spaan and Vlassis, 2004; Smith, 2007), some of which provide near-optimal solutions for problems with thousands of states. A state-of-the-art approach that involves sophisticated sampling of the belief space was proposed by Kurniawati et al. (2008), and was extended by Ong et al. (2009) to allow for mixed observability (e.g., when the locations of the searchers are completely known but the location of the target is unknown). The use of mixed observability extends POMDP capabilities to some problems with two robots.

These approaches provide probabilistic solutions to coordinated search problems, but they suffer from poor scalability because they plan in the joint space of searcher paths. In other words, the POMDP formulation requires a centralized planner to coordinate the paths of all searchers. In the multi-robot efficient search domain, the number of states in the POMDP formulation can easily reach several billion due to exponential scalability in the number of searchers. For instance, if one wanted to plan paths for six searchers looking for one target in an environment with 100 cells, a POMDP with $100^{6+1} = 10^{14}$ states would need to be solved. Solving POMDPs of this size is far outside the reach of even state-of-the-art POMDP solvers. The (PO)MDP formulation of coordinated search problems provides a fully probabilistic solution, but it suffers from these scalability issues. In contrast, the algorithms in this thesis utilize implicit coordination to remain tractable even for large numbers of searchers.

Probabilistic approaches using POMDPs have also been extended to deal with limited communication and team constraints. A number of approaches have shown how and showed how limited communication can be integrated into the Decentralized-POMDP (Dec-POMDP) framework (Nair et al., 2005; Roth, 2007; Carlin and Zilberstein, 2009). Emery-Montemerlo (2005) demonstrated how partially observable adversarial domains can be modeled with a Bayesian game approximation. While these approaches are applicable to a wide range of problems, they also suffer from poor scalability when faced with large problem instances. Thus, pure POMDP approaches are not sufficient to solve search problems with large teams in the physical world.

In addition, Hespanha and Prandini (2002) used a probabilistic framework to formulate a pursuit-evasion problem on partially known maps. They proposed a one-step greedy algorithm, and they proved that their algorithm generates a one-step Nash equilibrium (Hespanha et al., 2000). They test their algorithm only in simple environments, and they do not provide analysis of planning with a finite-horizon, which could improve performance over the greedy method. Finite-horizon planning allows for less myopic planning by enumerating possible strategies a number of steps into the future rather than simply to the next time step. Experiments in this thesis validate implicitly coordinated solutions and finite-horizon planning in partially known maps (see Chapter 5).

2.4 Combinatorial Optimization

Similar to fully probabilistic formulations, researchers have applied combinatorial optimization techniques to the coordinated search domain. Lau et al. (2006) presented a dynamic programming approach for efficiently finding a single non-adversarial target and a branch and bound approach for finding multiple targets (Lau et al., 2005). These techniques provide performance guarantees, but they suffer from poor scalability because the searchers plan in the joint space of searcher actions (i.e., they explicitly coordinate).

The poor scalability of probabilistic solutions to large environments and multiple searchers has motivated some researchers to utilize heuristic methods. Sarmiento et al. (2004) presented a framework for finding stationary targets in polygonal environments with multiple robotic searchers using a one-step cost heuristic. Instead of planning in the joint space, their algorithm reduces reward along previously traveled paths. This can be seen as an instance of implicit coordination. They do not extend their algorithm to mobile targets, and their work is purely heuristic without performance guarantees.

This thesis shows that the efficient search problem requires the optimization of a submodular objective function, and this key insight provides performance guarantees on implicitly coordinated solutions. Submodularity has been utilized in related domains to provide theoretical guarantees on similar algorithms. Guestrin et al. (2005) used submodular set functions to develop algorithms for sensor placement problems in Gaussian Processes and in more general domains (Krause and Guestrin, 2007). They prove that an implicitly coordinated algorithm provides the best possible guarantees in these domains, short of solving the NP-hard explicit coordination problem. These applications deal primarily with placing sensors to monitor information in an environment (e.g., monitoring algae blooms in lakes and temperature in a building), and they do not incorporate moving nodes (searchers). Singh et al. (2009a) developed algorithms for solving the multi-robot informative path planning (MIPP) problem, which do allow for moving nodes. However, their algorithms are not directly applicable to efficient search due the requirement that the starting and ending configurations be pre-specified.

Vidal et al. (2002) developed decentralized coordination strategies for a UAV/UGV team searching for multiple targets. They present two greedy coordination approaches, but they do not show performance guarantees on solution quality relative to optimal. They demonstrate empirically that their method is able to locate targets attempting to evade capture, but these evaders are only given limited capabilities. They do not consider a worst-case target, and their method cannot guarantee capture if the model of the target's motion is inaccurate. Bourgault et al. (2003, 2006) examined the problem of locating a potentially moving, non-adversarial target where the target's predicted motion is modeled using a Bayesian filter. They apply decentralized data fusion to develop a fully distributed approach. They also examine several candidate utility functions and present an optimal search strategy for a single searcher. However, they do not provide performance guarantees for their algorithm with multiple pursuers. Tisdale et al. (2009) applied receding-horizon control with a variable horizon to a UAV search application where the target's position is modeled using a probabilistic filter. Their objective is to maximize the information gain given a new UAV sensor measurement, which yields performance guarantees due to the submodularity of the objective function. They achieve decentralization using a version of implicit coordination, but they do not extend their method to adversarial targets or to account for team constraints. In addition, the calculation of their filter estimation could be costly for non-linear measurements and complex target motion models.

It is important to note that related work in static sensor networks combines worst-case and average-case guarantees. Krause et al. (2008) presented the SATURATE algorithm, which guides the placement of stationary sensors against an adversarial agent. SATURATE's runtime scales linearly with the number of actions available to the adversary. For search problems the number of actions (paths) available to the target scales exponentially with the size of the graph. Thus, SATURATE is infeasible for all but the smallest search problems.

2.5 Market-Based Strategies

One popular technique to improve scalability in multi-agent domains is for robots to coordinate using synthetic "auctions". The auction-based methods are specifically designed to inject tight coordination into the problem in places where it is most useful (Gerkey and Mataric, 2002; Kalra, 2006). In other words, auction methods explicitly coordinate when it is particularly beneficial.

A specific example where explicit coordination can be beneficial is the "valley problem" (Kalra, 2006). In this scenario, reward is maximized if two robots move through a valley. An alternative is for both robots to move around the valley, which gives a small reward. The lowest reward, or even a sharp penalty, occurs if one robot moves through the valley and the other does not. Auction-based methods solve the valley problem by selectively searching explicitly coordinated plans. One robot finds the two-robot valley path and then auctions it to the other robot. The second robot sees the advantage of such a plan and accepts the auction. In contrast, an implicitly coordinated solution will not find the two-robot valley path, and both robots will decide to go around the valley. In many multi-robot problems, the valley problem is either nonexistent or exceedingly rare. The results in this thesis show that this tends to be the case in efficient search. On the other hand, the valley problem more often arises during guaranteed search and constrained search. To avoid the valley in these cases without using explicit coordination, either an informed preprocessing step can be added (see Chapter 4), or the constraints can be relaxed somewhat (see

Chapter 6).

A number of researchers have applied auction methods to multi-agent coordinated search domains. Kalra (2006) presented Hoplites, an algorithm that utilizes auction-based plan sharing to perform tightly coupled tasks. Hoplites allows searchers to coordinate actively by running auctions when they are presented with high-cost situations, and it provides a framework for incorporating team constraints, which she demonstrates in the constrained search domain. Hoplites depends on multi-robot auctions to generate good search plans, but it does not provide a mechanism for deciding when to hold an auction if it is not obvious. Particularly during efficient search, it is difficult to determine when robots should hold auctions. Setting a synthetic threshold is one option, but this leads to poor performance if the threshold is set incorrectly. Methods using implicit coordination, on the other hand, provide an alternative that does not require the overhead of auctions.

Gerkey et al. (2005) also developed a parallel stochastic hill-climbing method (PARISH) for small teams that is closely related to auction-based methods. Rather than using the market metaphor, they frame guaranteed search as a parallel optimization problem. Their algorithm dynamically forms teams of searchers that work together to solve tasks. Team formation and path generation are guided by a heuristic, which makes their algorithm's performance sensitive to the choice of heuristic. Regardless of the heuristic used, the algorithm requires explicit coordination within teams, which can lead to high computation in large environments. PARISH is compared to the proposed algorithms in this thesis in Chapters 4 and 5.

It is important to note that auction-based methods are particularly suited to high-level task or resource allocation problems. Zlot and Stentz (2006) provided an overview of the specific applications of auction-based task allocation. Domains mentioned include search and rescue, hazardous waste cleanup, planetary exploration, surveillance, and reconnaissance. Auction-based methods can reason about complex task trees and multiple levels of abstraction to allocate tasks for a heterogenous team. Algorithms using implicit coordination, such as those in this thesis, can be utilized to solve lower-level tasks while auctions are used to solve the high-level task allocation problem. For instance, in an urban search and rescue scenario, some robots may need to clear rubble while others search the building. Auction-based task allocation would properly allocate rubble clearers and searchers. The searchers could then use an implicitly coordinated algorithm to search the building. If rubble clearing is necessary for search, the market algorithm and the implicit coordination algorithm could be interleaved to solve the task. The development of such combined search algorithms is left to future work.

While auction-based algorithms are more scalable than joint planning approaches, they still rely on auctions and/or team formation, which can consume large amounts of communication bandwidth and planning time. In many problems, including efficient search, implicit coordination is sufficient to produce high-quality results, and the additional overhead of running auctions is unnecessary. Adding an informed preprocessing step before implicit coordination enables implicit coordination even for more tightly coordinated tasks. The guaranteed search algorithm in this thesis is an example of such an approach.

2.6 Surveillance and Exploration

Often there is overlap between coordinated search and the areas of surveillance and exploration. For instance, Hegazy (2004) proposed decentralized algorithms for performing surveillance in urban environments. He examines both the non-adversarial and adversarial case, and he provides decentralized algorithms to optimize a cost heuristic in both cases. He incorporates target motion models and considers multiple targets during surveillance. Hegazy's algorithms demonstrate the feasibility of performing robotic surveillance in realistic environments, but they are primarily concerned with continuously monitoring urban environments and are not directly applicable to coordinated search.

The problem of searching environments has also been examined together with exploration. Calisi et al. (2007) provided a solution to the single robot exploration and search problem using petri-nets. Their work provides a principled architecture for robotic mission planning during urban search and rescue operations. However, they formulate their algorithm for the case of a single searcher, and it is difficult to see how it could extend to the multi-searcher case.

Scaling up to multiple robots, researchers have examined the problem of deploying a large team of heterogenous robots. Howard et al. (2005) showed how to deploy a large team of autonomous robots in an office building. After deployment, the team remains in place and monitors the environment. This technique is feasible if a large number of inexpensive robots are available for search. The work in this thesis focuses on reducing the number of mobile agents necessary during search through implicit coordination and mobile search schedules.

2.7 Constrained Planning

The constrained search problem is closely related to the more general constrained planning problem. During constrained planning, robots must perform a primary goal while maintaining some secondary constraints. Secondary constraints may exist during planning even for a single robot. For instance, a robot may need to move across a confusing environment without losing track of its position. Roy et al. (1998) introduced coastal navigation techniques that plan to encounter enough landmarks along a path to maintain localization. They take a dynamic programming approach, which minimizes the expected entropy of the robot's location while moving toward the goal. Since the constraints on the robot's motion come from the environment, they are not dependent on other robots on the team. Thus, this problem does not suffer from scalability issues with large teams.

For a team of multiple robots, constraints may include maintaining a line-of-sight chain, remaining within communication range, and/or reducing congestion at choke points. These constraints are often difficult to satisfy because they are dependent on the actions of the entire team (or a sub-team). There has been significant research in distributed control of mobile networks to maintain *continual* connectivity for path planning tasks. This thesis proposes *periodic connectivity*, where robots must regain connectivity once every fixed interval to replan, share information, and check for faults.

In many cases, continual connectivity is considered hard or unbreakable. Zavlanos and Pappas (2008) proposed a gradient-based optimization along with a market-based link deletion protocol to move a team of robots to a goal while maintaining hard connectivity constraints. Michael et al. (2008) implemented this market-based method on a team of mobile robots in an uncluttered environment. Their work utilizes the gradient of the second smallest eigenvalue (Fiedler value) of the Laplacian to determine the connectivity of the network. If this value is positive, the network is connected, and the value increases as the network becomes more connected. Ongoing work by Yang et al. (2008) showed that the Fiedler value can be computed using distributed algorithms. A similar gradient-based algorithm was employed by Hsieh et al. (2008), which allowed for building the communication graph online.

The gradient-based approaches described above are inherently myopic in that they do not consider more than one-step in the future. This limitation makes them poorly suited for domains that allow for periodic connectivity. When planning using a one-step gradient, it is not obvious how to make a plan that becomes temporarily disconnected and later regains connectivity.

In some domains, the network connectivity constraints are considered to be soft, or breakable. Krause et al. (2006) examined the problem of static network placement with varying connectivity throughout the network. They measured connectedness by the expected number of retransmissions required to send a message between two points. Their proposed algorithm has theoretical guarantees and performs well in static networks, but there is no straightforward extension to mobile networks or periodic connectivity.

Connectivity constraints are closely related to more general capacity constraints on a network. Calliess and Gordon (2008) used an online learning approach related to market-based approaches to solve the multi-agent routing problem. Their approach is potentially applicable to periodic connectivity, but it can only handle certain types of constraints. In addition, poor predictability of the reward function in real-world domains would require additional work to incorporate into their framework.

Market-based strategies (discussed in greater detail earlier) have also been applied to the constrained exploration domain (Kalra, 2006). Robots hold auctions for joint plans when they are faced with an impending constraint violation. The auctions allow the robots to coordinate explicitly at the proper times and avoid the constraint violation. As described above, auctions can require significant communication bandwidth and computation in large teams, and it can be difficult to determine whether or not an action should be held.

Anisi et al. (2008) introduced the idea of recurrent connectivity, where robots regain connectivity only when they make observations. They apply their approach to a coverage problem, but they do not generalize to broader problems or provide a scalable, decentralized algorithm.

2.8 Estimation and Tracking

During efficient search, the non-adversarial assumption allows the searchers to estimate the target's location using a probability distribution. Properly utilizing this estimate along with a motion model can lead to improved search times. In many applications of interest, the target lacks reliable odometry to assist in estimation. However, noisy measurements of the target's location may be available. This thesis examines the use of non-line-of-sight ranging radios capable of providing range between nodes through walls (Multispectral Solutions, Inc., 2008; Liao et al., 2006). These sensors have non-linear noise characteristics and can be difficult to model.

If the locations of the radio nodes are initially unknown, the estimation problem is closely related to simultaneous localization and mapping (SLAM). Djugash et al. (2008) proposed a range-only SLAM method using Extended Kalman Filters (EKFs). This EKF approach projects range measurements into polar space and uses a multi-modal representation to avoid errors from poor initialization, ambiguities, and noise. Despite these improvements, EKFs handle outliers poorly because they require linearization, and they are prone to error when odometry is poor or nonexistent.

Researchers in the sensor network community have also worked on localizing networks containing both moving and stationary nodes. Priyantha et al. (2005) proposed a method for coordinating a mobile node to localize a range-only network. Their method does not use a probabilistic formulation, which makes it poorly suited for sensors with nonstandard noise models. Their application is also different in that they have control over the mobile node. Hu and Evans (2004) also discussed localizing sensor networks that contain moving nodes. Their work is innovative in that it shows that moving nodes can help localize a sensor network, but they do not directly apply their method to range-only data.

Recent research has explored the use of Gaussian Processes for modeling the noise characteristics of non-linear sensors. Ferris et al. (2006) looked at tracking humans in office environments using measurements of wireless signal strength. Schwaighofer et al. (2004) also applied Gaussian Processes with the Matern kernel function for localization using cellular phone signal strength. Ranging radio estimation has both advantages and disadvantages when compared to wireless signal strength (Ferris et al., 2006, 2007). Data from wireless signal strength are often erratic and can be affected by slight changes in the environment. Ranging radio measurements more closely follow the true range between nodes, which allows for a more accurate measurement model. However, wireless access points are already part of the infrastructure of many buildings. Radio nodes would need to be placed in a building before use in tracking.

Without reliable odometry, recreating a path from range measurements becomes a non-linear dimensionality reduction problem. Gaussian Process Latent Variable Models (GPLVMs) were introduced by Neil Lawrence as a probabilistic framework for non-linear dimensionality reduction (Lawrence, 2005; Lawrence and Quiñonero-Candela, 2006). GPLVMs were later extended by Wang et al. (2007) to incorporate dynamics with applications to human motion modeling. Modeling dynamics allows for the incorporation of simple motion models into the GPLVM framework. Ferris et al. (2007) applied GPLVMs to solve the problem of localization with wireless signal strength when training data is unavailable. Their algorithm takes advantage of the above tools in a target tracking scenario. Work by Hollinger et al. (2008) goes one step further by using the reconstructed path to map the locations of ranging radio beacons. This approach allows for ad hoc localization of both the target and the multi-robot team during search applications.

2.9 Comparison of Related Work

Table 2.1 divides the related work into the three major coordinated search sub-problems (efficient, guaranteed, and constrained search) for both a single robot and for multi-robot teams. These divisions demonstrate how work in this thesis fits with prior work in each area. Figure 2.1 gives a comparison of related work using several metrics. The comparison assigns '+' and '++' to work that satisfies each metric. This table demonstrates that no work satisfies all the metrics addressed in this thesis. The metrics are described below.

- **Efficient search:**
  '+' = could be extended to efficient search
  '++' = directly addresses efficient search

- **Guaranteed search:**
  '+' = could be extended to guaranteed search
  '++' = directly addresses guaranteed search

- **Physical environments:**
  '+' = polygonal environments
  '++' = a representation based on the physical world

- **Scalable to large teams:**
  '+' = sub-exponential scalability
  '++' = linear or near-linear scalability

Table 2.1: Table of related work in coordinated search divided into sub-areas. Work in this thesis is shown in bold.

	Single Robot	Multi-Robot
Constrained Search	Coastal navigation (Roy et al., 1998)	**Periodic connectivity (Chapter 6)** Vector field (Zavlanos and Pappas, 2008) Coverage (Anisi et al., 2008) Signal strength maps (Hsieh et al., 2008) Game theoretic (Calliess and Gordon, 2008) Market-based (Kalra, 2006)
Guaranteed Search	Limited visibility (Gerkey et al., 2006) Escape length (Adler et al., 2003) Visibility-based (LaValle et al., 1997)	**GSST (Chapter 4)** GRAPH-CLEAR (Kolling and Carpin, 2010) IGNS (Kehagias et al., 2009b) Treewidth approximation (Fraigniaud and Nisse, 2006) Genetic algorithm (Flocchini et al., 2005) PARISH (Gerkey et al., 2005) Randomized search (Isler et al., 2005) Search on trees (Barrière et al., 2002)
Efficient Search	Petri-nets (Calisi et al., 2007) Dynamic prog. (Lau et al., 2006) Branch and bound (Lau et al., 2005) POMDP (Roy et al., 2005)	**FHPE+SA (Chapter 3)** MOMDP (Ong et al., 2009) UAV coordination (Tisdale et al., 2009) Submodular path planning (Singh et al., 2009a) Limited communication (Emery-Montemerlo, 2005) Stationary target (Sarmiento et al., 2004) Decentralized Bayesian search (Bourgault et al., 2003) UAV/UGV coordination (Vidal et al., 2002) Unknown environments (Hespanha and Prandini, 2002)

- **Decentralized:**
  '+' = capable of running without a central coordinator

- **Models target motion:**
  '+' = motion modeling assists search

- **Handles sensor uncertainty:**
  '+' = capable of modeling sensor noise

- **Team constraints:**
  '+' = incorporates constraints between team members

- **Limited communication:**
  '+' = capable of operating under communication limitations

- **Partially known environments:**
  '+' = capable of operating without full knowledge of the environment map

Algorithms in this thesis satisfy the criteria above, but they suffer from some limitations. Implicit coordination requires that searchers know their own locations to some degree of certainty and can understand the relative locations of their teammates. This restriction precludes the use of robots with minimal awareness. Other extensions not examined in this thesis include: cooperative targets, self-interested searchers,

	Efficient search	Guaranteed search	Physical environ.	Scalable to large teams	Models target motion	Handles sensor uncertainty	Decentral.	Team constraints	Limited comm.	Partially known environ.
(Anisi et al., 2008)			+	+				+		
(Barrière et al., 2002)		++		++			+			
(Bourgault et al., 2003)	+		+	++	+	+	+		+	
(Calisi et al., 2007)	+		++							+
(Calliess and Gordon, 2008)	+			+	+		+	+		+
(Emery-Montemerlo, 2005)	+		++		+	+	+	+	+	
(Flocchini et al., 2005)		++		++			+			+
(Fraigniaud et al., 2006)		++		+						
(Gerkey et al., 2005)		++	++	+	+		+	+		
(Hegazy, 2004)			++	+	+	+	+			
(Hespanha et al., 2002)	++			+	+	+				+
(Howard et al., 2005)			++	+		+	+	+	+	
(Hsieh et al., 2008)	+		++	++		+	+	+		
(Isler et al., 2005)		++		+						
(Kalra, 2006)	+	+	++	+			+	+	+	
(Kehagias, 2009b)		++	++	+			+			
(Kolling and Carpin, 2010)		++	++	+		+				
(Lau et al., 2005, 2006)	++		++		+					
(LaValle et al., 1997)		++	+							
(Ong et al., 2009)	++		++		+	+				
(Roy et al., 2005)	+		+	+	+	+				
(Sarmiento et al., 2004)	++		+	++		+	+			
(Singh et al., 2009)	+	+	++	+		+	+			
(Tisdale et al., 2009)	+		++	++	+	+	+			
(Vidal et al., 2002)	++		++	+	+	+	+			+
(Zavlanos et al., 2008)				+			+	+		
This Thesis	++	++	++	++	+	+	+	+	+	+

Figure 2.1: Comparison of related work. Pluses denote that the work addresses each metric. Double pluses denote that the work handles a metric particularly well.

searchers with large differences in capabilities, and task allocation when search is one among many tasks.

Furthermore, this thesis focuses directly on multi-robot search problems. Implicit coordination methods are applicable to problems other than multi-robot search, but determining a good representation can be problem specific and often nontrivial. This thesis provides the groundwork for the development of a generalizable implicit coordination framework for a broad spectrum of multi-agent problems. Specific avenues for future work towards this goal are described in more detail in Chapter 9.

Chapter 3

Efficient Search

THIS chapter examines the efficient search problem of locating a mobile, non-adversarial target using multiple robotic searchers. If the searchers have an idea of how the target moves, they can utilize this information to improve their estimate of the target's location and the efficiency of the search. The use of such an approximate model of the target's behavior can lead to faster search times than more conservative assumptions (e.g., assuming the worst-case behavior of the target). The problem becomes one of choosing searcher paths most likely to intersect with the path taken by the target.

Despite the formal hardness of finding the optimal solution, near-optimal performance can be achieved without explicitly coordinating the actions of the team. This chapter presents an algorithm that allows each team member to plan its own actions and implicitly coordinate by sharing that information, which provides a decentralized and high-performing solution to the efficient search problem. In addition, theoretical analysis in this chapter shows that the diminishing returns property of the objective function, formalized by submodularity, leads to approximation guarantees on the implicitly coordinated solution, which underly it high performance.

The proposed approximation algorithm utilizes replanning on a fixed horizon and implicit coordination to achieve an online solution that is linearly scalable in the number of searchers. It is proven that solving the efficient search problem requires maximizing a nondecreasing, submodular objective function, which leads to theoretical bounds on the performance of the proposed algorithm. The analysis is extended by considering the scenario where searchers are given noisy non-line-of-sight ranging measurements to the target. For such a scenario, online Bayesian measurement updating is integrated into the framework. The performance of the proposed framework is demonstrated in two large-scale simulated environments, and further validated using data from a novel ultra-wideband ranging sensor. Results show that the proposed linearly scalable approximation algorithm generates searcher paths competitive with those generated by exponential algorithms.[1]

3.1 Efficient Search Problem Setup

This section formally defines the problem of locating a mobile, non-adversarial target with multiple searchers, which will be referred to as the Multi-robot Efficient Search Path Planning (MESPP) problem. It is also shown that the MESPP problem optimizes a submodular objective function. To formulate the MESPP problem, the environment in which the searchers and target are located needs to be described. First, we divide the environment into convex cells (see algorithm description below). Taking into account the cell adjacency in a discretized map yields an undirected graph that the searchers can traverse. Let $G = (N, E)$ be the undirected environment graph with vertices N and edges E. At any time t, a searcher exists on vertex $s(t) \in N$. The searcher's movement is deterministically controlled, and it may travel to vertex $s(t + 1)$ if there exists an edge between $s(t)$ and $s(t + 1)$. A target also exists on this graph on vertex $e(t) \in N$. The target moves probabilistically between vertices. The searcher receives reward by moving onto the same vertex as the target, $s(t) = e(t)$, and no reward is gained after this occurs (this is referred to as a capture event). Reward is discounted by γ^t, where γ is a constant discount factor. Thus, the searcher receives more reward for finding the target at a lower t. The discount factor corresponds to the probability that the search will end at a given time. For instance, the target may leave the search area or expire, which necessitates locating the target in a short time.

It is assumed that the searcher has an approximate model of the target's motion that is both Markovian (i.e., the target's next move depends solely on its current cell) and independent of the searcher's position on the graph. This assumption allows for a rich space of motion models, including those followed by randomly moving and stationary targets. If a perfect model of the target's motion is not known, any number of approximate Markovian models could be used, including a general random model. The searcher knows its own position, and it has knowledge of the target's position at time t in the form a belief distribution $b(t)$ over all vertices. Since $b(t)$ can be an arbitrary distribution, this formulation allows multi-modal estimates of the target's position. Let us call the problem so far the Efficient Search Path Planning (ESPP) problem.

To extend to MESPP, we place K searchers on the vertices; the location of the k-th searcher at time t is $s_k(t) \in N$ (for $k = 1, 2, ..., K$ and $t = 1, 2, ..., T$). The searchers now gain reward if any of them are on the same vertex as the target. Incorporating additional searchers forces both the state and action space to grow exponentially. Let us refer to the combined action space of all searchers as the *joint* action space (see Chapter 1 for a formal definition). The MESPP problem (as well

[1]The algorithm in this section was originally introduced by Hollinger et al. (2009b).

as the ESPP problem) can be formulated as a Partially Observable Markov Decision Process (POMDP) with the reward function below:

$$J\left(U(1), \ldots, U(T)\right) = \sum_{t=0}^{T} \gamma^t \Pr(\exists k : s_k(t) = e(t)), \tag{3.1}$$

where $U(1), \ldots, U(T)$ are the deterministic *actions*, with $U(t) = [U_1(t), \ldots, U_K(t)]$ being a K-dimensional control vector specifying the location of each searcher (i.e., $s_k(t) = U_k(t)$). The goal is to choose $U(1), \ldots, U(T)$ that maximizes $J(U(1), \ldots, U(T))$ as given by Equation 3.1.

As shown below, the reward function J is both *nondecreasing* and *submodular* on the set of nodes of the *time-augmented search graph*. The meaning of these terms is as follows:

1. The time-augmented search graph G' is a *directed graph* and is obtained from G as follows: if u is a node of G then (u, t) is a node of G', where $t = 1, 2, ..., T$ is the *time stamp*; if uv is an edge of G, then $(u, t)\,(v, t+1)$ and $(v, t)\,(u, t+1)$ are *directed* edges of G' for every t. There is also a directed edge from (u, t) to $(u, t+1)$ for all u and t. In other words, G' is a "time evolving" version of G and every path in G' is a "time-unfolded" path in G.

2. A set function is nondecreasing if adding more nodes to the observed set always increases reward. This is clearly the case in MESPP since visiting more places can only increase likelihood of capture.

   Definition A function $F : \mathbf{P}(N') \to \Re_0^+$ is called *nondecreasing* iff for all $A, B \in \mathbf{P}(N')$, we have

   $$A \subseteq B \Rightarrow F(A) \leq F(B).$$

3. A set function is submodular if it satisfies the notion of diminishing returns. In other words, the more places in the environment that have been visited, the less can be gained by visiting more.

   Definition A function $F : \mathbf{P}(N') \to \Re_0^+$ is called *submodular* iff for all $A, B \in \mathbf{P}(N')$ and all *singletons* $C = \{(m, t)\} \in \mathbf{P}\left(N'\right)$, we have

   $$A \subseteq B \Rightarrow F(A \cup C) - F(A) \geq F(B \cup C) - F(B).$$

Searchers choose feasible paths that maximize the expected probability of intersecting the target's path at the earliest possible time (before reward is heavily discounted). Any such path for searcher k visits a set of nodes in G'; call it $A_k \subseteq G'$. The union of searcher paths is referred to as $A = A_1 \cup \ldots \cup A_K$. A encodes both which nodes of the original G have been visited and at which times.[2] Hence the reward $J\left(U(1),\ldots,U(T)\right)$ can also be written as $J\left(A\right)$. Now we write $J\left(A\right)$ in more detail as follows: let Y denote a path (in G') taken by the target, let $\Pr\left(Y\right)$ be the probability of this path, let Ψ be the space of all possible target paths; finally, let $F_Y(A)$ be the discounted reward received by searcher paths A if the target chooses path Y. This gives us the reward function in Equation 3.2.

$$J(A) = \sum_{Y\in\Psi} \Pr(Y) F_Y(A) \tag{3.2}$$

We now have a theorem concerning the properties of $J\left(U(1),\ldots,U(T)\right) = J\left(A\right)$.

Theorem 3.1 *The objective function optimized by the MESPP problem is $J(A)$, and this is a nondecreasing, submodular set function.*

The proof of Theorem 3.1 is given in the Appendix. This result is used in the next section to show bounds on the performance of sequential allocation in the efficient search domain. Since the Markov assumption is made on the target's motion model, calculating the expectation in Equation 3.2 can be done using dispersion matrices. This simplifies the computation of $F(A)$ over the space of searcher paths, as explained in more detail below.

3.2 Environment and Target Modeling

3.2.1 Map Discretization

The formulation of the efficient search problem requires the discretization of continuous environments into discrete cells. Partitioning the environment into cells allows for planning on a finite graph and greatly reduces the complexity of planning and estimation. One method for discretization takes advantage of the inherent characteristics of indoor environments. To discretize an indoor map by hand, simply label convex hallways and rooms as cells and arbitrarily collapse overlapping sections. This method is simple enough that it can be performed by hand even for large maps. Alternatively, a suitable discretization can be found automatically using a convex region finding algorithm (see Chapter 5). Taking into account the cell adjacency in a discretized map yields an undirected graph that the searchers can traverse. Figure 3.1 shows an example discretization of a small house environment and the resulting undirected graph. Subsequent sections refer to this simple example for explanatory purposes.

The convexity of the cells guarantees that a searcher in a given cell will have line-of-sight to a target in the same cell. Gaining line-of-sight is relevant to most sensors

[2]Note that this formulation assumes that multiple searchers can visit the same node at the same time. The time-stamped nodes visited by each searcher also define its path. For ease of notation, we will sometimes refer to A_k as a searcher path and A as a set of paths.

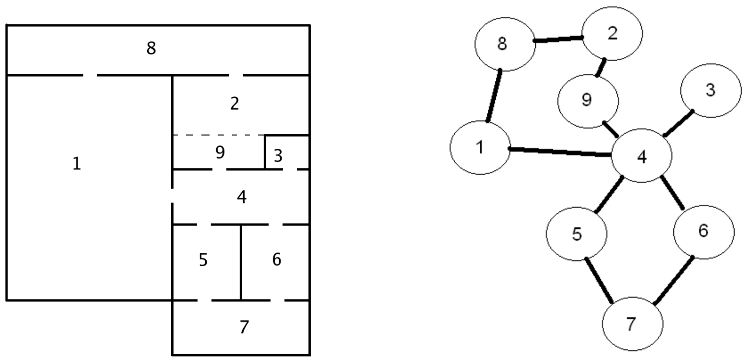


Figure 3.1: Example discretization of house environment (left) and undirected graph resulting from the house discretization (right). Searchers plan paths on the undirected graph that maximize capture probability of a non-adversarial target.

that a mobile robot would carry, including cameras and laser scanners. Convex cells allow for search schedules with a 180 degree field-of-view sensor. The requirements are that: (1) a searcher can see an entire cell while at a boundary (pointing inward), and (2) the searcher can move from one cell border to another while keeping the entire destination border in view at all times (Gerkey et al., 2005).

Figure 3.2 shows example discretizations of the office building and museum environments used for simulated testing. Both of these environments are larger and more complex than those discussed in previous work in coordinated search (Guibas et al., 1999; Lau et al., 2005; Sarmiento et al., 2004). The museum environment is particularly challenging because it contains many cycles by which the target can avoid detection. In contrast, the office environment has two major cycles that correspond to the hallways. Several even larger and more complex environments are considered in Chapter 5.

Alternatively, a suitable discretization can be found automatically by generating a constrained Delaunay triangulation (CDT) of the environment (Shewchuk, 2002). Given a set of vertices and edges (the obstacle and environment boundaries), the CDT is the triangulation of the vertices with the following properties: (1) the prespecified edges are included in the triangulation, and (2) it is as close as possible to the Delaunay triangulation. The CDT ensures that the edges of the obstacles correspond to edges in the triangulation, which generates a convex tessellation of the free space. The resulting search graph has a bounded branching factor of three because it consists only of triangles. Delaunay triangulations can be generated with angle and size requirements on the polygons, which allows for the inclusion of range constraints on the sensors. If a triangular tessellation is not suitable, any convex tessellation can be used, but we note that the relationship between the discretization and the number of searchers required is not well understood. In some cases, a larger number of cells may actually require fewer searchers, which leaves open the question

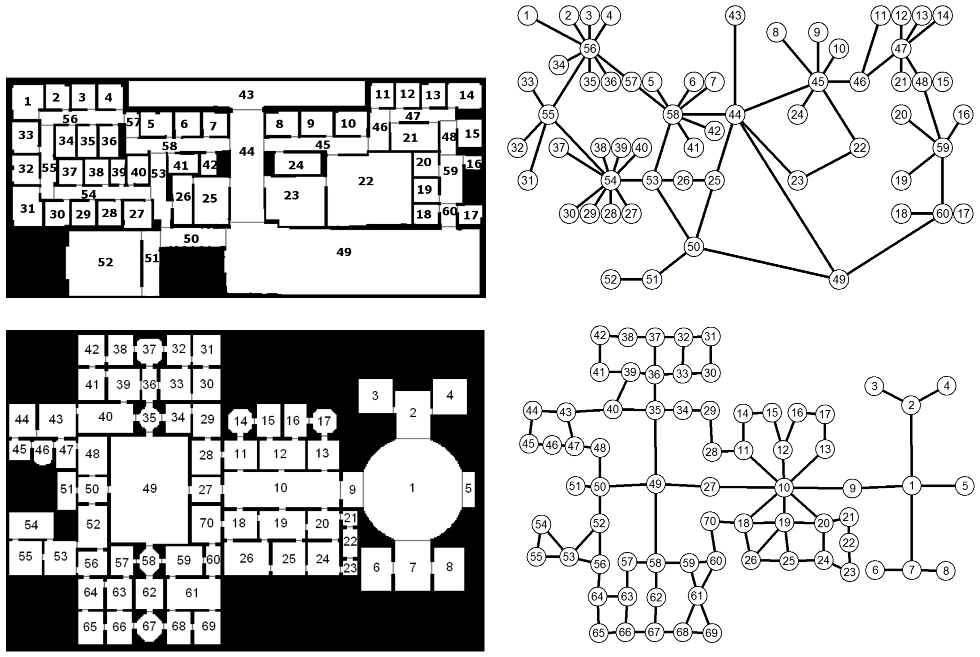


Figure 3.2: Example floorplans and graphical representation of environments used for search trials. The Newell-Simon Hall (top) and National Gallery of Art (bottom) were used for simulated testing.

of how best to discretize to help minimize the number of searchers.

In indoor environments, there are several different categories of "obstacles" that may exist. In this thesis, major obstacles (e.g., walls and buildings) are included in the tessellation, but minor obstacles, such as small furniture, are not. It is assumed assume that the robots' sensors can see through minor obstacles, or that the target is large enough that it could not hide behind these obstacles. Of course, a finer discretization taking into account minor obstacles can also be obtained and utilized, at the cost of a more complex environment graph. The types of obstacles to consider during the discretization in various environments is a design decision and a subject of ongoing research.

3.2.2 Target Motion Modeling

To integrate a motion model of the target into the efficient search framework, "capture" and "dispersion" matrices can be applied to the target's state vector. This formulation simplifies the calculation of Equation 3.2 and its optimization.

The location of the target is represented by a belief vector $b(t) = [b_0(t), \ldots, b_N(t)]$ where $b_0(t)$ represents the probability the target has been captured by time t (the capture state), and $b_n(t)$ represents the probability that at time t the target is in the n-th discretized cell. We can mathematically represent a capture event on that state vector by defining a matrix that moves all probability from all cells visible from searcher k's current cell $s_k(t)$ to the capture state. The capture matrix can also contain non-unity values if the probability of seeing a target when it is in a searcher's line-of-site is less than one. The appropriate capture matrix $C_{s_k(t)}$ for cell $s_k(t)$ is applied at time t as in Equation 3.3.

$$b(t+1) = b(t)C_{s_k(t)} \tag{3.3}$$

For example, if it is assumed that the searcher cannot see through doorways, the capture matrix for a searcher in cell one would be the $(N+1) \times (N+1)$ identity matrix with the second row unity value shifted to the first column.

Similarly, we can define dispersion matrices to represent the expected motion of the target in the environment. The discretization of the environment yields an undirected graph of possible target movements between cells. Based on a motion model, probabilities are assigned to each of these movements, which define a matrix that properly disperses the target's probable location.

Utilizing a dispersion matrix implicitly assumes that the target's motion model obeys the Markov Property, which is equivalent to saying that the probability of the target moving to a given cell is dependent solely on the target's current cell. This assumption allows for the representation of many motion models including random and stationary. It also allows for the target's motion model to be modified based on the size of the cells or the probability that the target will move through certain areas of the map.

The dispersion matrix formulation also has the advantage that the target's probability distribution is fairly insensitive to changes in the motion model. Since the transition probabilities are solely dependent on the current state, changing the probability of moving between cells slightly will not cause large changes in the probability distribution. The dispersion matrix D at time t can be applied to yield a new target state vector at time $t+1$ as in Equation 3.4. If a stationary model for the target is preferred, D can be set to the identity.

$$b(t+1) = b(t)D \tag{3.4}$$

For instance, if it is assumed that the target will remain in its current cell or move to any adjacent cell with equal probability at the next time step, the dispersion matrix for the house environment in Figure 3.1 is given below. Note that this is only one of the many motion models expressible by dispersion matrices. The top row of the dispersion matrix corresponds to the capture state. For reference, row 5 (emboldened) corresponds to the probabilities associated with cell 4. If the target's cell transitions were not equally likely, the row values would no longer be equal. It is important to note that while there are as many capture matrices as there are cells in an environment, there is a single dispersion matrix for the entire environment.

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \mathbf{0} & \boldsymbol{\frac{1}{6}} & \mathbf{0} & \boldsymbol{\frac{1}{6}} & \boldsymbol{\frac{1}{6}} & \boldsymbol{\frac{1}{6}} & \boldsymbol{\frac{1}{6}} & \mathbf{0} & \mathbf{0} & \boldsymbol{\frac{1}{6}} \\ 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & 0 & 0 & 0 & 0 & 0 & \frac{1}{3} & 0 \\ 0 & 0 & \frac{1}{3} & 0 & \frac{1}{3} & 0 & 0 & 0 & 0 & \frac{1}{3} \end{pmatrix}$$

The capture matrix of a searcher k and the dispersion matrix can be multiplied yielding a new target state vector as in Equation 3.5.

$$b(t+1) = b(t)DC_{s_k(t)} \tag{3.5}$$

In larger environments, it is desirable to use multiple searchers. The capture matrices for all searchers can be multiplied to yield the new state as in Equation 3.6. To perform this step, it is necessary for the searchers to communicate their states at each time step.

$$b(t+1) = b(t)D[\prod_{k=1}^{K} C_{s_k(t)}], \tag{3.6}$$

where K is the number of searchers.

A state evolution equation can now be written for the system of searchers and target. The state vector at time t is $S(t) = [s_1(t), \ldots, s_K(t), b_0(t), \ldots, b_N(t)]$, and the state evolution equation has the form:

$$S(t+1) = f(S(t), U(t+1)), \tag{3.7}$$

where $U(t{+}1) = [U_1(t{+}1), \ldots, U_K(t{+}1)]$ is a K-dimensional control vector specifying the next location of each searcher (as described in Section 3.1), and the evolution of the probabilities $b_0\,(t+1)\,, \ldots, b_N\,(t+1)$ is determined by Equation 3.6.

Recall that the reward function is:

$$J(U(1), \ldots, U(T)) = \sum_{t=0}^{T} \gamma^t \Pr(\exists i : s_i(t) = e(t)), \tag{3.8}$$

where T is some finite ending time. Hence, the MESPP optimization problem is to maximize Equation 3.8 subject to the state evolution shown in Equation 3.7.[3]

[3]Also recall that Equation 3.1 can be used as the reward function in a POMDP formulation of

The computational complexity of the dispersion and capture matrix application is determined by the number of cells in the environment (the size of the matrices) and the number of searchers. It is $O(K|N|^3)$, where K is the number of searchers and $|N|$ is the number of cells in the environment. While $O(|N|^3)$ may grow intractable for very large numbers of cells, many competing methods scale exponentially in either $|N|$ or K (e.g., polygonal search using visibility graphs (Guibas et al., 1999)). In addition, the matrices are often quite sparse, which reduces the complexity of applying the dispersion and capture matrices.

3.2.3 Measurement Incorporation

The efficient search formulation assumes that the target is found when the searchers achieve line-of-sight to it. If some cases, noisy or ambiguous non-line-of-sight measurements of the target's location are available, which the searchers can use to improve the search. For example, ranging measurements may be used to provide a circular annulus of possible target locations. Such measurements can improve the estimate of the target's position, but they are not accurate enough to provide a capture event. After receiving a new measurement, searchers can replan using the new belief distribution on the target's location. Since searchers replan after receiving measurements, online measurement incorporation heuristically improves path quality. The next section shows how combining sequential allocation, finite-horizon path enumeration, and Bayesian measurement updating yields a scalable and online algorithm for solving the MESPP problem.

3.3 Coordination Methods

3.3.1 Explicit Coordination

Having properly modeled the target's motion, the searchers can look for paths on the discretized floorplan with high reward. One method is to use a centralized planner to solve the full POMDP for the infinite horizon, but finding such a solution becomes intractable very quickly. An alternative is to search all possible joint paths on the floorplan graph to a given depth, which outputs paths for all searchers that maximize reward on the finite-horizon. Searching the joint path space can be decentralized by assigning identification numbers to the searchers and forcing each searcher to plan as if it were the centralized solver (the searchers must still communicate their state). We refer to this method as explicit coordination because each searcher explicitly plans for its teammates.

The advantage of explicit coordination is that each searcher takes the future positions of the other searchers into account during planning. However, the dimensionality

MESPP.

of the search space grows such that explicit coordination scales exponentially with the number of searchers. The number of cells that must be searched is $O(b^{dK})$, where K is the number of searchers, d is the lookahead depth, and b is the maximum branching factor of the cell graph. Such a high complexity algorithm does not lead to a tractable solution for more than a small number of searchers with a short lookahead.

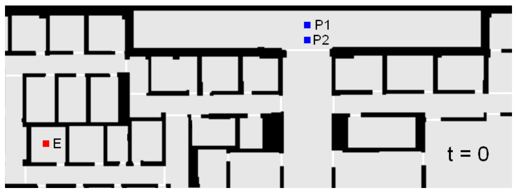
3.3.2 No Coordination

To decouple planning during efficient search, each searcher can plan for itself while assuming that the states of the other searchers are fixed. Decoupling prevents the search space from growing in complexity as the number of searchers increases. With this assumption, each searcher must simply plan for its optimal path given the current state information of the other searchers. The complexity of this planning algorithm *on each searcher* is not affected by the number of searchers: $O(b^d)$. Even though each searcher's planning is independent of its teammates' future actions, the current positions of other searchers provide information that the target is not in that cell. For this purpose, searchers must communicate their locations at each time step.

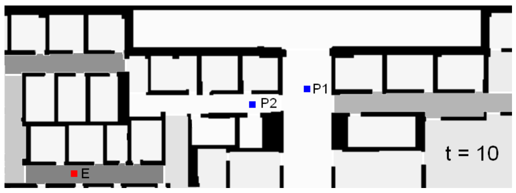
3.3.3 Implicit Coordination

Without coordination, the searchers cannot reason about the future actions of their teammates. To overcome this disadvantage, a coordination method can be introduced that is intermediate between explicit coordination and no coordination at all. During implicit coordination, the searchers share their current paths after planning. Other searchers then plan for their own paths while assuming that the transmitted paths will be followed. This strategy leads to higher quality solutions than the no coordination strategy with the same scalability on each searcher: $O(b^d)$. However, to guarantee that all robots have the current paths of the rest of the team, planning must be done sequentially, which requires $O(Kb^d)$ computation and is linear in the number of searchers. In some scenarios, this linear increase in planning time can be avoided through an extension of implicit coordination where each agent replans at different points on the map (see Section 3.5).

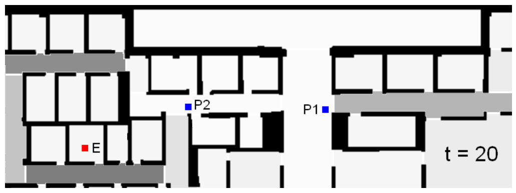
The major cost of implicit coordination is that searchers must now communicate their entire paths rather than just their current locations. Since the environment is coarsely discretized, sharing paths does not lead to much increase in communication bandwidth. However, implicit coordination does lead to a broadcast communication requirement, which can delay communication in large networks. In this case searchers may need to plan with outdated information. This problem is alleviated by the sparsity of the network because searchers will typically only communicate with teammates in close proximity, which gives them up-to-date information from searchers near them and somewhat outdated information from searchers far away. Since planning is up to a finite horizon, information from near searchers is more im-



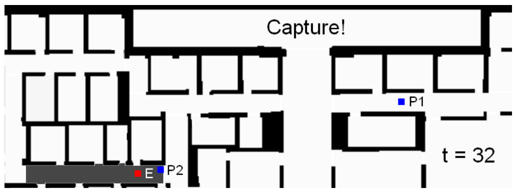
(a) Time = 0 s



(b) Time = 10 s



(c) Time = 20 s



(d) Time = 32 s

Figure 3.3: Snapshots of implicit coordination during coordinated search at different time steps until capture event. The searchers (labeled P1 and P2) branch into the two major cycles on the map to search for the target (labeled E). Darker cells denote more probable target locations.

portant. Thus, network delay will have minimal affect on solution quality in sparse networks. In addition, communication complexity can be reduced using heuristics, such as broadcasting paths only to searchers within the horizon length. Limited communication is analyzed in more detail in Chapter 6.

Figure 3.3 shows how implicit coordination can lead to high quality searcher paths. In this example, the searchers correctly branch into the two major cycles on the map. Searcher two communicates its intention to move left to searcher one. Searcher one then uses that information to choose the path to the right. Generalizing this approach leads to an implicit coordination algorithm utilizing sequential allocation. The searchers plan one-at-a-time and share their paths with the other searchers. Each searcher takes into account information from their teammates but can only modify its own path.

3.4 FHPE+SA Algorithm

This section describes an algorithm for non-adversarial search with multiple robots utilizing sequential allocation and finite-horizon planning. Theoretical performance bounds are shown using the nondecreasing submodularity of the MESPP objective function. This somewhat surprising result shows that sequential allocation, an algorithm linearly scalable in the number of searchers, generates near-optimal paths in the MESPP domain.

Algorithm 3.1 Sequential allocation MESPP algorithm

```
Input: Multi-agent efficient search problem
% $V \subseteq N'$ is the set of nodes to be visited by searchers
$V \leftarrow \emptyset$
for all searchers $k$ do
    % $A_k \subset N'$ is a feasible path for searcher $k$
    % Finding this argmax solves the ESPP for searcher $k$
    $A_k^* \leftarrow \text{argmax}_{A_k} F(V \cup A_k)$
    $V \leftarrow V \cup A_k^*$
end for
Return $A_k^*$ for all searchers $k$
```

3.4.1 Sequential Allocation

Algorithm 3.1 gives pseudocode for the MESPP sequential allocation algorithm. The algorithm maintains a list of nodes $V \subseteq N'$ that have been visited by the searchers. Note that N' is the time-augmented version of the nodes N in the environment, which allows for nodes in the original graph to be revisited at later times. Each searcher chooses a path A_k that maximizes the objective function $F(V \cup A_k)$ and then adds the nodes they have visited to V. Effectively, subsequent searchers treat the paths of previous searchers as "given", and they are not allowed to change them. Sharing nodes to update V is an instance of implicit coordination as described above. Since the search space does not grow with the number of searchers, the complexity of sequential allocation is linear in the number of searchers. It is important to note that while path planning occurs sequentially, the execution of paths is simultaneous.

Algorithm 3.1 requires maximizing the objective function for the ESPP problem as a subroutine. Any algorithm for solving the ESPP problem can be inserted here. However, if the ESPP solver is bounded with respect to optimal, the nondecreasing submodularity of the objective function leads to guarantees on the performance of sequential allocation. A theorem from previous work shows that sequential allocation leads to theoretical guarantees in the related informative path planning domain.

Theorem from Singh et al. (2009a): *Let κ be the approximation guarantee for the single path instance of the informative path planning problem for any nondecreasing, submodular function. Then sequential allocation achieves an approximation guarantee of $(1 + \kappa)$ for the multi-robot informative path planning problem.*

The findings in Theorem 3.1 can be leveraged to extend these results to MESPP. Corollary 3.1 states that if an ESPP solver has an approximation guarantee of κ, then sequential allocation on the MESPP will yield an approximation guarantee of $(1+\kappa)$.

Corollary 3.1 *If a solver achieves an approximation guarantee of κ for the ESPP problem, sequential allocation yields an approximation guarantee of $(1 + \kappa)$ for the Multi-robot ESPP (MESPP) problem.*

Algorithm 3.2 Finite-horizon path enumeration (FHPE) for ESPP

```
Input: Single-agent efficient search problem
for All feasible paths $A$ to horizon $d$ do
    Calculate $F(A)$
end for
Return $A^* \leftarrow \arg\max_A F(A)$
```

Proof The proof of Corollary 3.1 is immediate from Singh et al. (2009a) and Theorem 3.1. Singh et al. (2009a) show that sequential allocation achieves this bound for any single-agent path planning problem optimizing a nondecreasing, submodular function. Theorem 3.1 shows that the ESPP problem requires the optimization of such an objective function. ∎

Here an approximation guarantee κ states that if the MESPP solver returns a set of nodes $A \subseteq N'$, then $F(A) \geq \frac{1}{\kappa} F(A^{OPT})$, where A^{OPT} is the set of nodes visited by the optimal paths. Clearly, $\kappa \geq 1$ since $F(A)$ cannot be greater than the optimal reward. The case where $\kappa = 1$ corresponds to solving the ESPP problem optimally. In this case, sequential allocation can achieve no worse than half the optimal reward. This theoretical result allows single-agent performance bounds to be extended to the multi-agent case using a linearly scalable algorithm, albeit with a loss in approximation quality. The next section presents a bounded algorithm for solving the ESPP problem using finite-horizon path enumeration.

3.4.2 Finite-Horizon Planning

In large environments, even the single-agent ESPP problem may be intractable to solve optimally (or even near-optimally) due to the computational overhead of considering many infinite-horizon paths. In these cases, one option is for the searchers to plan a finite number of cells ahead and choose the best path to that horizon. At any time while traversing this path, the searcher can plan again utilizing new information on a new horizon. Receding-horizon planning leads to an online solution to MESPP, and it allows for the incorporation of measurements of the target's position as they become available. Algorithm 3.2 gives pseudocode for solving the ESPP problem using finite-horizon path enumeration with sequential allocation (FHPE+SA). Since the finite-horizon method relies on path enumeration, it scales exponentially with the search depth.

Lemma 3.1 derives performance guarantees for finite-horizon path enumeration, and the result extends to the multi-robot case with sequential allocation as in Theorem 3.2.

Lemma 3.1 *Finite-horizon path enumeration on the ESPP problem achieves a lower bound of:*

$$F(A^{FH}) \geq F(A^{OPT}) - \epsilon, \tag{3.9}$$

where A^{FH} is the path returned by finite-horizon path enumeration, A^{OPT} is the optimal feasible path, and $\epsilon = R\gamma^{d+1}$.

Proof Finite-horizon path enumeration achieves the optimal reward inside the horizon depth for ESPP because it checks all paths. The maximum reward that could be gained outside the horizon depth is given by $\epsilon = R\gamma^{d+1}$, where R is the arbitrarily selected reward received for locating the target, γ is the discount factor, and d is the search depth. The bound is immediate. ▮

Theorem 3.2 *Finite-horizon path enumeration with sequential allocation (FHPE+SA) on the K-robot MESPP problem achieves a lower bound of:*

$$F(A_1^{FH} \cup \ldots \cup A_K^{FH}) \geq \frac{F(A_1^{OPT} \cup \ldots \cup A_K^{OPT}) - \epsilon}{2} \tag{3.10}$$

Proof The maximum reward outside the horizon remains the same as in Lemma 3.1 (i.e., ϵ is unchanged). Since the single robot case achieves the optimal reward within the finite-horizon (i.e., $\kappa = 1$), sequential allocation yields an approximation guarantee as in Corollary 3.1 as $\kappa + 1 = 2$. ▮

Intuitively, as search depth increases, the bound tightens. Additionally, decreasing the discount factor tightens the bound. This is because smaller discount factors more heavily weight reward gained earlier, which is more likely to be within the finite-horizon. It is important to note that the quality of the bound is independent of the number of searchers. This is a worst-case bound for arbitrary starting distributions and motion models. In practice, searchers can run finite-horizon path enumeration repeatedly on a receding horizon, which leads to performance that far exceeds this lower bound.

3.4.3 Bayesian Measurement Updating

This section presents a discrete Bayesian method for modifying the target's state probability vector given new sensor data. Suppose at time t a measurement $\mathbf{z}_t$ is received. We are interested in computing the *conditional* probability $p_t^i = \Pr(e_t = i|\mathbf{z}_1, \ldots, \mathbf{z}_t)$.[4]

In other words, p_t^i is the probability that the target is in node i at time t, conditioned on measurements $\mathbf{z}_1, ..., \mathbf{z}_t$. Note that in previous sections we were computing $\Pr(e_t = i)$, i.e. the probability of the target being at node i without any measurement conditioning. A known motion model is also given, which provides $\Pr(e_t = i|e_{t-1} = j)$ for all cells i and j. The motion model is encoded in the dispersion matrix D.

Using standard recursive Bayesian updating (Thrun et al., 2005), a measurement update can be written as in Equation 3.11. Note that if a measurement is not received at time t, the motion model could simply be applied at that time step.

[4]Note: the target's location is denoted as e_t instead of $e(t)$ for this section to simplify notation.

$$p_t^i = \eta \Pr(\mathbf{z}_t | e_t = i) \sum_j \Pr(e_t = i | e_{t-1} = j) p_{t-1}^j, \tag{3.11}$$

where η is a normalizing constant.

If the target's motion model is known, the problem of calculating the posterior is reduced to that of calculating a likelihood term $\Pr(\mathbf{z}_t|e_t = i)$. Since each cell is represented as a continuous set of points in the map plane, this calculation is difficult. To reduce the complexity of the problem, we further discretize each cell into small subcells and calculate a likelihood at the center of each subcell. The M^i subcells of cell i are denoted as m^{ij} for all $j \in \{1, \ldots, M^i\}$. The calculation of $\Pr(\mathbf{z}_t|e_t = i)$ is now one of calculating a likelihood at many points and then taking the sum of these likelihoods.

For range measurements, the problem of calculating $\Pr(\mathbf{z}_t|e_t = i)$ is simply that of determining the expected range value for the center of each subcell. Let q^{ij} be the Euclidean distance from the ranging sensor to subcell m^{ij}, and let r_t be the received range measurement with assumed Gaussian noise variance σ^2. The likelihood is then calculated as in Equation 3.12.

$$\Pr(\mathbf{z}_t | e_t = i) = \sum_{j=1}^{M^i} N(r_t; q^{ij}, \sigma^2) \tag{3.12}$$

To improve accuracy using the discrete Bayesian method, an element-based discretization method can also be incorporated (Furukawa et al., 2007). Element-based methods use irregular polygonal discretizations to interpolate the posterior between points. The techniques in this chapter allow for this extension if greater accuracy is necessary than that provided by a regular grid.

The framework proposed in this chapter has been extended to scenarios in which the locations of the radio nodes are initially unknown (similar to a simultaneous localization and mapping scenario). Hollinger et al. (2008) use probabilistic dimensionality reduction with Gaussian Process Latent Variable Models (GPLVMs) to reconstruct a preliminary path of the target. The path is then used to map the locations of the radio nodes on an occupancy grid. The reconstructed node locations can be used for subsequent tracking. This approach provides a measurement incorporation framework for efficient search in environments without pre-installed infrastructure.

3.5 Efficient Search Experiments

3.5.1 Simulated Experiments

To test the proposed MESPP algorithm, simulated trials were run using a multi-agent coordinated search simulation in C++ on a 3.2 GHz Pentium 4 processor. This simulation allows for multiple searchers and both stationary and moving targets.

It was assumed that the average speed of the target is 1 m/s, and that it moves holonomically between cell boundaries. The searchers also move with a maximum speed of 1 m/s, which would be a reasonable speed for state-of-the-art autonomous vehicles. The searchers start together in the same location for all trials, and the location of the target is initialized at random on the map. Simulated experiments were run in the museum (150 m $\times$ 100 m) and office (100 m $\times$ 50 m) environments shown in Figure 3.2.

In the following trials, searchers wait for the entire team to reach their replanning points, and then planning is performed sequentially as described in Algorithm 3.1. This implementation allows for ease of comparison with POMDP methods and the validation of theoretical claims. An extension that allows for asynchronous planning is shown in Section 3.5.3. The performance metric in this section is the average reward received over many trials. For a given trial, reward received is calculated as $R(t_c) = R\gamma^{t_c}$, where R is the reward for locating the target, γ is the discount factor, and t_c is the number of steps taken on the graph before the target was found. The reward and discount factor were arbitrarily set to $R = 1$ and $\gamma = 0.95$ for all experimental trials.

The first experiments validate the finite-horizon approximation. Trials were run with a single searcher with an increasing horizon depth. An increase in reward of approximately 5% was seen in the office and 10% in the museum when the horizon was increased from one to two. Further horizon increases resulted in very little increase in reward up to a horizon of six. Horizons greater than six became computationally intensive (see Section 3.5.4). Figure 3.4 compares FHPE (horizon depth five) to the infinite horizon POMDP solution. With a single searcher, the POMDP formulation of ESPP is still solvable using Heuristic Search Value Iteration (HSVI2) (Smith, 2007). HSVI produces a near-optimal solution that is guaranteed to be within ϵ of the optimal. For these trials, $\epsilon = 0.1$. These results show that, for the single searcher case, finite-horizon path enumeration yields average rewards competitive with those generated by the HSVI POMDP solution. Trials with two searchers were attempted with HSVI but were unsuccessful because the exponentially increased state-action space would not fit in memory. The failure to find a solution with two searchers demonstrates the relatively poor scalability of the POMDP formulation of the MESPP problem.

FHPE+SA is also compared to three widely used general POMDP heuristic solvers in Figure 3.4. The most likely heuristic (Nourbakhsh et al., 1995) takes the highest probability state where the target may be and returns the best possible team action if that were its true location. The voting heuristic (Simmons and Koenig, 1995) allows each possible target state to "vote" for the next team action, and votes are weighted by the belief that the target is in that state. The QMDP heuristic (Littman et al., 1995) solves the underlying MDP for each possible target state and then uses the value function to weight the voting for the next team action.

One somewhat surprising result is that the most likely heuristic outperforms

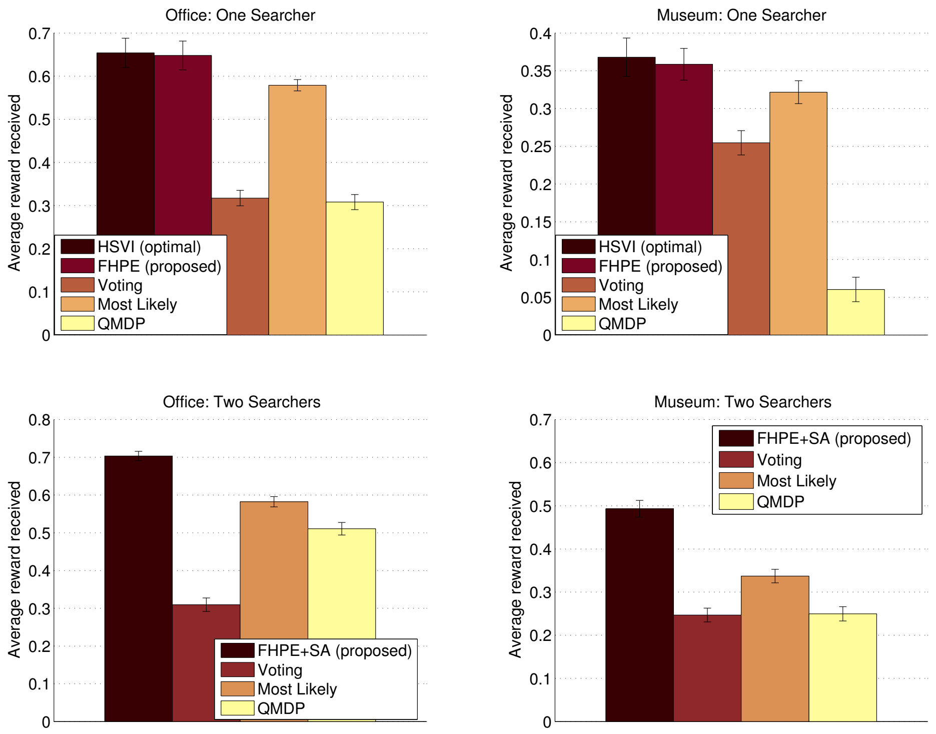


Figure 3.4: Comparison of FHPE+SA (horizon depth five) with HSVI (near-optimal solution) and several general POMDP heuristics in two simulated environments. The target is moving randomly in these trials. HSVI ran out of memory on the two searcher problem before generating a solution. Error bars are one standard error of the mean (SEM), and averages are over 200 trials.

QMDP is this domain, which is counter to results in many other domains (Littman et al., 1995). One explanation is that QMDP leads to policies that are too conservative. The QMDP policies put the searchers in a position that minimizes the expected distance to the target if its position were to become known at the next time step. However, the searchers never quite reach the highest probability cells. In contrast, the most likely heuristic, which selects the highest probability cell and moves towards it, is more effective, particularly when the searchers take different paths to the most likely cell. All of these heuristics are essentially myopic in that they do not account for gaining information about the targets position. Thus, it is not surprising that the proposed algorithm outperforms the general heuristics in both environments (see Figure 3.4).

Since solving the POMDP formulation is intractable for multiple searchers, trials were run using a finite-horizon explicit coordination algorithm that is identical to Algorithm 3.2 except that it enumerates paths for all searchers in the joint space. Finite-horizon explicit coordination scales $O(b^{dK})$, where b is the branching factor, d

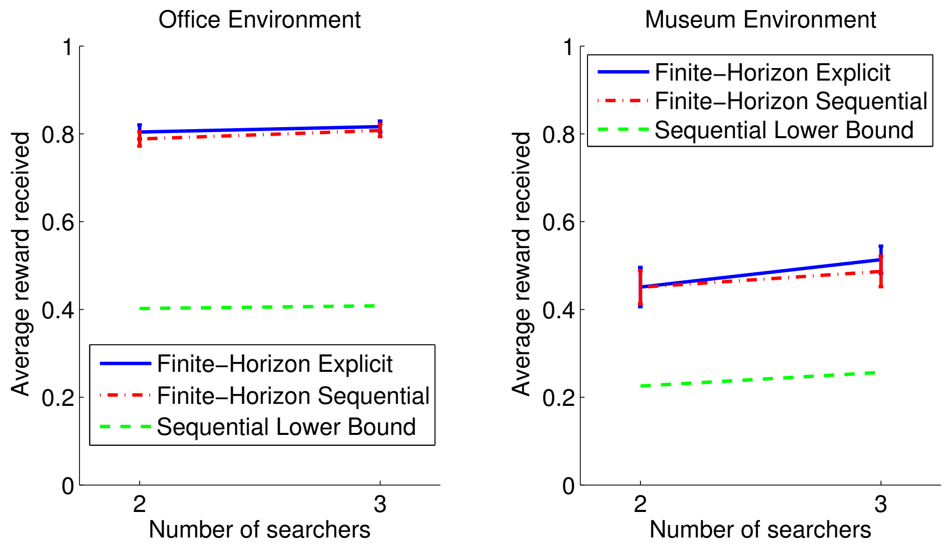


Figure 3.5: Comparison of sequential allocation versus explicit coordination in two simulated environments. FHPE+SA with horizon two was used to allow for direct comparison, since explicit coordination with a horizon greater than two was infeasible. Error bars are one standard error of the mean (SEM), and averages are over 200 trials. Target and searchers move at a maximum speed of 1 m/s. Sequential allocation greatly outperforms its lower bound.

is the search depth, and K is the number of searchers. Figure 3.5 gives a comparison of reward received by sequential allocation and this explicit coordination algorithm. Since explicit coordination exceeds computational limits at large horizon depths, a depth of two was used for comparison. Explicit coordination is also infeasible with more than two searchers. Figure 3.5 also shows a lower bound for sequential allocation calculated from Corollary 3.1 using the explicit coordination results. The bound represents the lowest reward that sequential allocation could achieve if explicit coordination yielded the optimal reward. On both maps, sequential allocation greatly outperforms its lower bound.

These simulated experiments demonstrate that implicit coordination with sequential allocation yields results nearly equivalent to those achieved through explicit coordination. In sharp contrast with explicit coordination's exponential scalability, sequential allocation is linearly scalable in the number of searchers. Figure 3.6 demonstrates the scalability of sequential allocation by showing reward received with up to five searchers in the museum and office.

3.5.2 Ranging Radio Measurements

One major application of MESPP is that of finding lost first responders in disaster scenarios. To better model this scenario, an urban response test environment was set

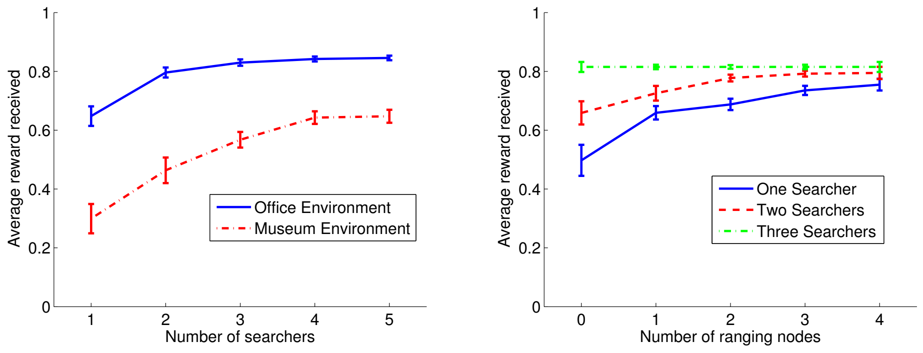
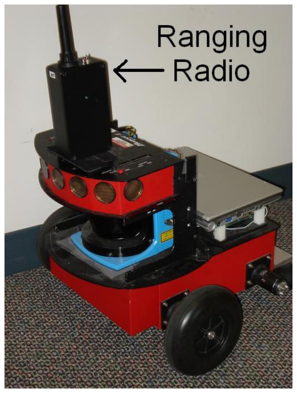


Figure 3.6: Multiple searcher scalability trials for FHPE+SA (horizon five) in simulated environments (left) and with ranging radio nodes in a real-world environment (right). Error bars are one standard error of the mean (SEM), and averages are over 200 trials. Target and searchers move at a maximum speed of 1 m/s in the simulated trials and 0.3 m/s in the real-world trials. The zero node case in the real-world trials corresponds to the absence of radio measurements.

up using a Pioneer robot and five Multispectral ranging radio nodes (Multispectral Solutions, Inc., 2008). These sensors use ultra-wideband signals to provide inter-node ranging measurements through walls. They have an effective operating distance of about 30 m indoors and provide ranging accuracy approximately within 1 m. In experiments, the Pioneer robot acted as a lost first responder and was teleoperated around the environment carrying a ranging radio node. Four stationary nodes were placed in surveyed locations around the environment to provide range to the Pioneer. The Pioneer also carried a SICK laser scanner, and its location was found using laser Adaptive Monte-Carlo Localization methods from the Carmen software package (Thrun et al., 2005). The Pioneer's laser localization was used for ground truth, but was not used to assist in search. The Pioneer's maximum speed was set to 0.3 m/s, the maximum that provided consistent laser localization. Figure 3.7 shows a photograph of the Pioneer robot as well as the office environment used for testing.

After gathering data from the ultra-wideband ranging sensors, simulated searchers were added to the environment. These searchers have access to the ranging measurements from the stationary nodes in the environment, which allows them to utilize real range data from the experiment to find the target in the simulated world. The searchers were given a maximum speed of 0.3 m/s to match that of the Pioneer target. The recursive Bayesian measurement incorporation method was used in these trials. Computational limits allowed for cell sizes of approximately 10 cm $\times$ 10 cm.

Figure 3.8 shows the results for one and two searchers in these "hybrid" trials. As in purely simulated trials, the finite-horizon path enumeration method provides nearly equivalent reward as the POMDP solution. Sequential allocation is competitive with explicit coordination in these results as well. These results show that the measurement incorporation framework is effective with real data from ultra-wideband



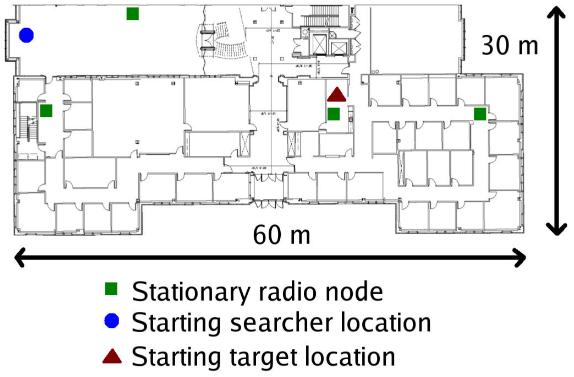
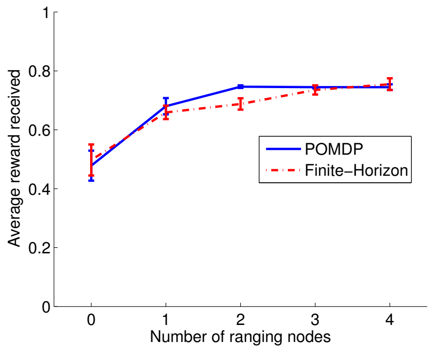


Figure 3.7: Photograph of Multispectral ultra-wideband ranging radio mounted on Pioneer robot (left) and floorplan of testing environment (right). The robot was teleoperated around the environment to act as the moving target.



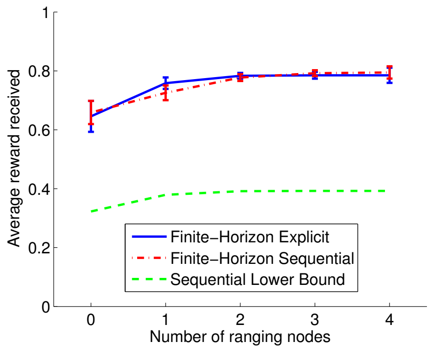


Figure 3.8: Left: Comparison of FHPE (horizon depth five) versus the POMDP solution for a single searcher using ultra-wideband ranging radio measurements from experimental trials. Right: Comparison of sequential allocation versus explicit coordination with two searchers using ranging radio measurements. As in simulated trials, sequential allocation greatly outperforms its lower bound. Both: Error bars are one standard error of the mean (SEM), and the searchers move at a maximum speed of 0.3 m/s.

ranging sensors.

Since the experiments are run in playback, the number of sensors used can be varied by turning off some sensors' data streams. Figure 3.6 shows average rewards using an increasing number of ranging radio nodes. The zero node case corresponds to search without non-line-of-sight measurements. The results show that adding more searchers leads to decreasing capture times. Increasing the number of measurement beacons also leads to decreasing capture times. These results suggest that if a small number of searchers are available, this can be compensated with more measurement beacons, and vice versa. In fact, adding additional searchers more effective than adding measurements, to the point where a single searcher cannot achieve performance competitive with two searchers even when using all measurement beacons. This tendency is likely due to the noisy data received from the ranging radios. If the measurements were less noisy, they would improve performance more. In this environment, adding more than three searchers no longer improves performance.[5]

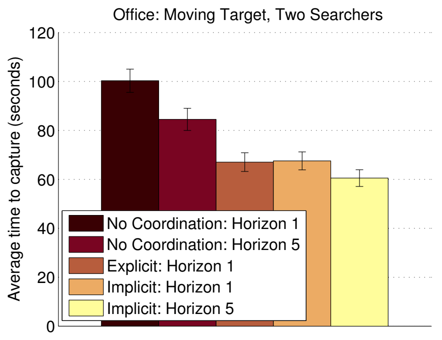
3.5.3 Asynchronous Operation and Capture Time

In real-world scenarios, it is beneficial if each searcher does not need to wait for the rest of the team to finish planning before beginning its own replanning. Waiting for teammates leads to long wait times and poor performance when the distances between cells are highly non-uniform. An extension of FHPE+SA allows searchers to plan sequentially initially and then replan whenever they reach the centroid of a cell. Each searcher utilizes the current version of its teammates' plans when replanning (i.e., each searcher assumes that all other searchers will properly execute their shared plans when planning its own path). Thus, searchers do not need to wait for other searchers before replanning. This implementation allows for asynchronous planning and also maintains many of the benefits of sequential planning. In the following results, asynchronous operation is allowed, and planning time is considered in the final average capture time (see Section 3.5.4 for more analysis of planning time).

The previous results use the discounted reward metric, which is an important metric because it relates the efficient search problem to similar problems in probabilistic planning. In real scenarios, however, average capture time may be a more relevant metric. This section presents results showing that the proposed algorithm reduces average capture time as well as discounted reward. Figure 3.9 shows a comparison of different coordination strategies and planning horizons in the museum and office using the average capture time metric. The results show that myopic planning (one-step lookahead) does not perform as well as using a five-cell lookahead. Further simulated testing (not shown), suggests little improvement in average capture times with lookahead greater than five cells. The results also show that implicit coordination provides low average capture times in these environments. Due to its poor scalability, explicit coordination could not be run with a five-step lookahead. However, the improved scalability of implicit coordination allows for non-myopic planning and the lowest capture times.

For further comparison, Figure 3.10 shows average capture times for the proposed method (without non-line-of-sight measurements) against a random search strategy. In the random search strategy, the searchers randomly move between cells in the environments. The proposed method using implicit coordination and finite-horizon path enumeration yields nearly a factor of five improvement over the random strategy in many cases. In addition, a randomly moving target is much easier to capture in the NSH environment than a stationary target. This result is due to the large number

[5]Multimedia extensions (see Appendix A) show playback of FHPE+SA search schedules from experiments with and without ranging radio measurements.



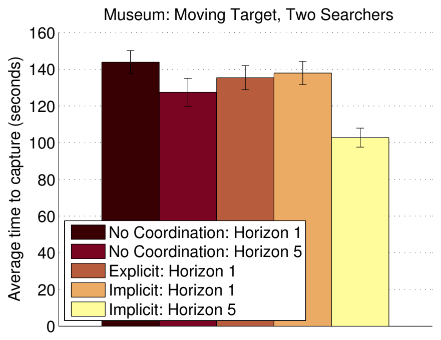


Figure 3.9: Comparison of different coordination methods during efficient search. Results show that implicit coordination yield capture times competitive with explicit coordination. Two searchers were used in all trials. Error bars are one SEM, and averages are over 200 trials. Target and searchers move at a maximum speed of 1 m/s.

of rooms connected to the hallways in the NSH environment. The searchers can stay in the hallways and have a very high chance of catching a moving target. In contrast, they must search the rooms and then backtrack to the hallways to locate a stationary target. The higher connectivity of the museum negates this effect because the searchers no longer need to backtrack out of the rooms, and the strategy for the moving target becomes similar to that of a stationary target.

3.5.4 Planning Time

This section examines the planning time required by implicit coordination using both the synchronous (sequential) and the asynchronous planning strategies. Figure 3.11 shows average planning time for an increasingly large team size. The average planning time is the time to generate a single replan for the entire team. The planning time scales linearly with the number of searchers, but it remains below 0.1 seconds with five searchers. The average number of replans before capture are also shown. The total planning time, number of replans times average planning time, remains on the order of a few seconds even with the highest number of replans.

If planning is performed asynchronously when searchers reach specified replanning points, the planning time no longer increases with the number of searchers. Each searcher plans its own path, and applies capture matrices corresponding to the shared paths of its teammates during planning. Table 3.1 shows computation times for a single robot planning on a varying horizon as it searches the museum and office environments. Even with a horizon of six, planning typically takes less than one-fourth of a second, which would not affect real-time operation. A searcher could begin planning a second or more before reaching its goal and then continue without stopping.

A potential problem with asynchronous operation is that race conditions can occur

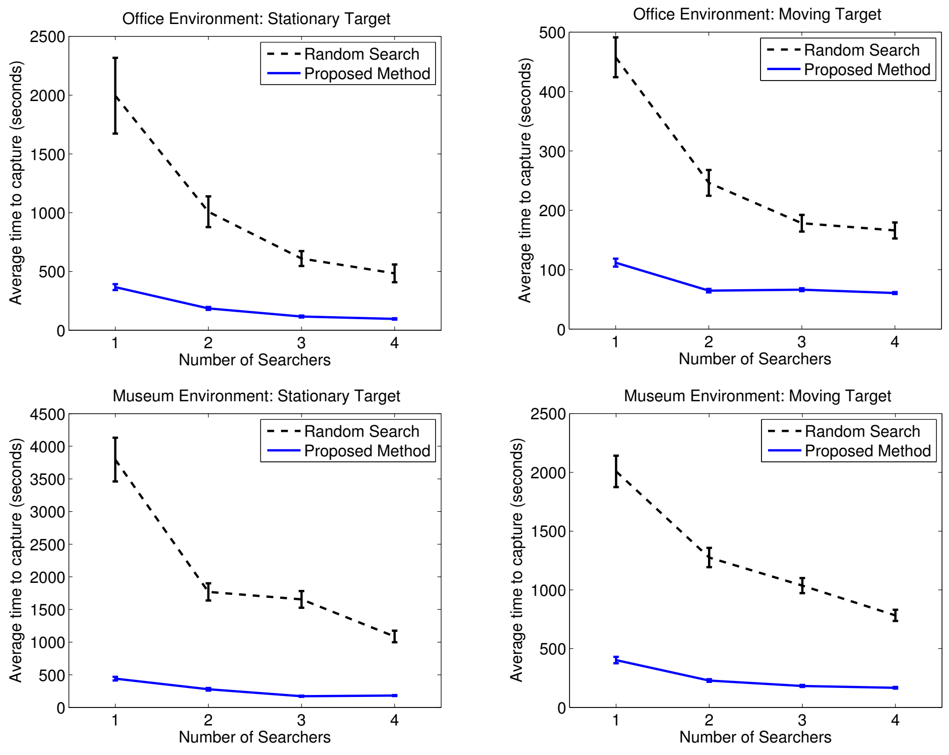


Figure 3.10: Simulated coordinated search results comparing the proposed method to random search. The proposed method uses implicit coordination without non-line-of-sight measurements. Graphs show average capture times versus number of searchers for a stationary target (left) and a moving target (right). Error bars are one SEM, and averages are over 200 trials. Target and searchers move at a maximum speed of 1 m/s.

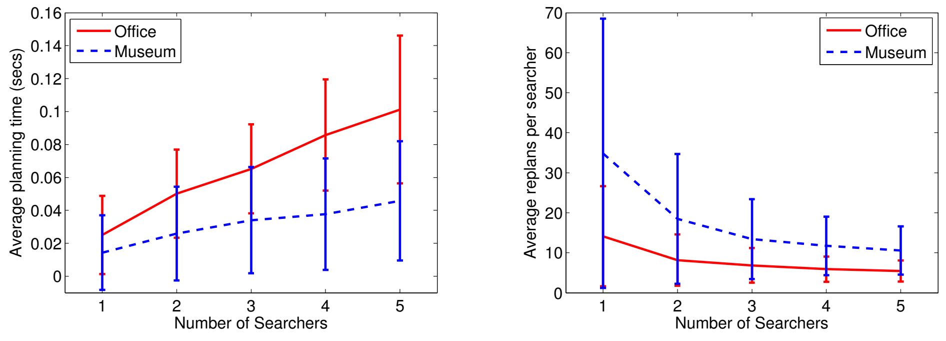


Figure 3.11: Average planning time for FHPE+SA with synchronous sequential planning (left). Error bars are one standard deviation. The planning time scales linearly with the team size. Average replans required per searcher to find a randomly moving target (right). Even with the highest number of replans, the total planning time is less than two seconds.

Table 3.1: Average planning times for varying horizon lengths for a single searcher using FHPE in the office and museum maps. Errors are one standard deviation. Note that the office has a higher average branching factor than the museum, and hence larger planning times. Using a horizon greater than six, the planning time becomes greater than several seconds, which would hinder real-time operation.

Horizon	$d = 1$	$d = 2$	$d = 3$
Office	20 ± 17 μsec	29 ± 9 μsec	216 ± 580 μsec
Museum	15 ± 153 μsec	27 ± 11 μsec	177 ± 554 μsec
Horizon	$d = 4$	$d = 5$	$d = 6$
Office	1.38 ± 1.34 msec	24.4 ± 23.1 msec	247 ± 191 msec
Museum	1.05 ± 1.69 msec	14.1 ± 2.23 msec	187 ± 315 msec

if planning and communication are not instantaneous. However, these situations were rare in this domain due to the fact that robots would rarely reach replanning points at the same time. Furthermore, planning time is nearly negligible compared to execution time, as shown in Table 3.1 and Figure 3.11. Communication limitations are explored in more detail in Chapter 6, where all-to-all communication is no longer assumed.

3.6 Chapter Summary

This chapter presented FHPE+SA, a scalable algorithm for solving the Multi-robot Efficient Search Path Planning (MESPP) problem of locating a non-adversarial target using multiple robotics searchers. The MESPP problem was formally defined and shown to be consistent with the Partially Observable Markov Decision Process (POMDP) framework. It was shown that current POMDP solvers are incapable of handling large instances of MESPP. The proposed algorithm uses sequential allocation and finite-horizon path enumeration to remain computationally tractable for multiple searchers in large environments. In addition, this chapter derived performance guarantees for sequential allocation by exploiting the nondecreasing submodularity of the MESPP objective function. Sequential allocation is an instance of implicit coordination during which multiple robots share information rather than planning in the joint path space. Sequential allocation is linearly scalable, and it remains tractable in large problem instances when exponential methods using explicit coordination are far beyond computational limits. The simulated and experimental results using ultra-wideband ranging radios show the performance of the proposed algorithm in complex environments. An asynchronous version of the algorithm was also shown to perform well when the robots replan at designated points in the environment given their current available information.

The results in this chapter demonstrate that a simple sequential planning approach yields theoretically bounded and experimentally high-performing results in the efficient search domain. The theoretical guarantees stem from the diminishing returns property of the objective function, which can be linked to a loose coupling between the actions of the team members. In other words, the success of one robot's plan does not depend heavily on the success of the plans of its teammates. If a robot fails to execute its plan, the results are not catastrophic. The team plan suffers somewhat since reward is lost by the failure, but the remaining team can still achieve a high reward. In domains that require tighter coordination, it may be necessary for a number of teammates to complete their plans to achieve any reward at all. This requirement clearly breaks the diminishing returns property of the objective function.

Throughout this chapter, it was assumed that the target's motion model was non-adversarial and could be approximated by the searchers. These assumptions lead to the diminishing returns property mentioned above and the theoretical bounds on the performance of sequential allocation. If the target is acting adversarially or the searchers' model is not representative of the target's actual behavior, the diminishing returns no longer hold, and the performance of the algorithm is no longer bounded. In these cases, searchers running FHPE+SA can fail to find the target entirely. The following chapters show how utilizing an underlying spanning tree to guide the search can extend the advantages of implicit coordination to domains with worst-case adversarial targets. Ultimately, this approach leads to decentralized, online, and high-performing solutions in domains requiring tighter coordination.

Chapter 4

Guaranteed Search

THIS chapter examines the guaranteed search problem of coordinating multiple robots such that a target cannot escape detection. This situation arises in at least two cases. The first is if the target is acting adversarially, and the second is if an accurate motion model of the target is unavailable. In both cases, the searchers wish to guarantee that the target will be found regardless of its movement pattern. In contrast with efficient search, which seeks to exploit a motion model to maximize capture probability, guaranteed search makes a worst-case assumption on the target's behavior.

Guaranteed search increases the coupling between the searchers' actions, since multiple searchers must properly execute their plans to clear the environment and complete the task successfully. In domains requiring tight coordination, straightforward application of implicit coordination often performs poorly. However, it is shown that utilizing an informed environment representation allows for high-performing implicitly coordinated strategies even in the tightly coupled guaranteed search domain.

The problem discussed in this chapter is as follows: clear a graphical representation of the environment of any adversarial target using the fewest number of searchers. This problem is NP-hard on arbitrary graphs but can be solved in linear-time on trees. This chapter presents Guaranteed Search with Spanning Trees (GSST), an anytime algorithm for multi-robot search. GSST utilizes the optimal schedule from an underlying spanning tree of the environment to guide the search. At any time, spanning tree generation can be stopped, yielding the best schedule so far. The resulting schedule can be modified online if additional information becomes available. Though GSST does not have performance guarantees after its first iteration, several variations will find an optimal solution given sufficient runtime. GSST is tested in simulation and on a human-robot search team using a distributed implementation; it quickly generates clearing schedules with as few as 50% of the searchers used by competing algorithms in the literature.[1]

[1]The GSST algorithm was originally introduced by Hollinger et al. (2010a).

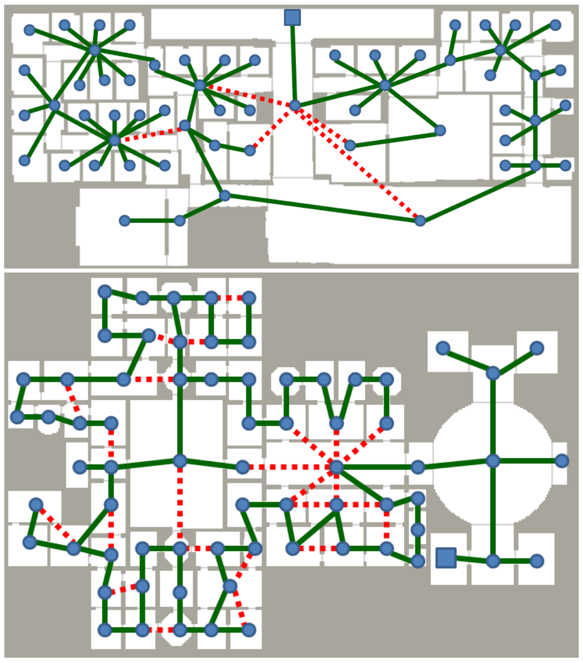


Figure 4.1: Spanning trees generated by the GSST algorithm are used to clear a graphical representation of a physical environment of any potential adversary. The searchers utilize the optimal search schedule on an underlying tree to guide the search of the graph. Guards are used to prevent recontamination on edges not in the spanning tree. The office (top) admits a search schedule with three searchers, and the museum (bottom) admits one with five. Green (solid) lines denote edges in the spanning tree, and red (dashed) lines denote edges in the graph but not in the tree. The square denotes the searchers' starting position.

GSST utilizes implicit coordination on an informed environment representation to achieve better scalability than centralized and market-based approaches. Implicit coordination can provide poor solutions in domains like guaranteed search that require tight coordination. To improve performance for these problems, searchers can execute a shared preprocessing step, which transforms the environment representation into one solvable through implicit coordination. Utilizing spanning trees of the environment yields implicitly coordinated search schedules for guaranteed search (see Figure 4.1 for examples). This approach leads to an "anytime" algorithm for guaranteed search, which quickly finds a solution with potentially many searchers and then continues to generate solutions with fewer searchers over time. This technique yields guaranteed search paths with few searchers even in large environments.

In addition to its anytime capabilities, GSST can be easily decentralized, and it allows for some modifications of the initial schedule to be performed during execution. Example modifications include dealing with the case of fortuitous information (e.g., an area of the environment happens to be cleared during the schedule), and correcting for poorly synchronized movement between the searchers. These capabilities are demonstrated on a team of human-robot searchers in an indoor urban search and rescue scenario.

4.1 Guaranteed Search Problem Setup

This section formally defines the guaranteed search problem on a graphical representation of a physical environment. As in the previous chapter, let $G = (N, E)$ be the undirected environment graph with vertices N and edges E. This formulation requires discretization into a number of cells. As in the previous chapter, this requirement is accomplished by dividing the environment into a convex tessellation (see Chapter 3).

At any time t, the k-th of K searchers exists on vertex $s_k(t) = u \in N$. The searcher's movement is deterministically controlled, and each may travel to vertex $s_k(t+1) = v$ if there exists an edge between u and v. A target also exists on this graph on vertex $e(t) = u \in N$. The target moves along edges between vertices. A searcher "captures" the target by moving onto the same vertex (i.e., $\exists k, t : s_k(t) = e(t)$). It is assumed in this chapter that a searcher on a given node will always detect a target on the same node, and the target is assumed to have potentially unbounded speed.

At any time during the search, there are nodes that may contain the target (dirty nodes) and nodes that may not (cleared nodes). The set of dirty nodes at time t is denoted by $N_D(t) \subseteq N$ and the cleared set by $N_C(t) = N \setminus N_D(t)$. The searchers' goal is progressively to decrease the set of dirty nodes to the empty set, thus guaranteeing capture of any target in the environment.

Definition The cleared set $N_C(t) \subseteq N$ is the set of nodes that can no longer contain the target at time t. Conversely the dirty set $N_D(t) \subseteq N$ is the set of nodes that may still contain the target. $N = N_C(t) \cup N_D(t)$, and the two sets are mutually exclusive, which implies $N_C(t) = N \setminus N_D(t)$ for all t.

Definition A schedule $\mathbf{S} \subseteq N'$ (i.e., a feasible set of node/time pairs visited by the searchers) is a clearing schedule if the dirty set $N_D(t_f) = \emptyset$ at some time t_f (which also implies $N_D(t) = \emptyset$ for every $t \geq t_f$).

Definition The node search number $s(G)$ of a graph G is the minimum number of searchers required to generate a clearing schedule on that graph.

Given the definitions above, the recontamination rules must be defined to formulate the guaranteed search problem. It is assumed that the target exists on the nodes of the graph, and that it cannot move through nodes containing searchers. In addition, the target may be arbitrarily fast, and it is has complete knowledge of the searchers' strategy. Thus, any node in the cleared set that does not contain a searcher and is adjacent to a node in the dirty set will be recontaminated and switch to the dirty set.

Definition A node v is considered recontaminated at time t if $v \in N_C(t-1)$, v does not contain a searcher at time t, and there exists a path that does not contain a searcher to a node $u \in N_D(t)$. If these conditions are true then $v \in N_D(t)$.

The problem discussed in this chapter is to generate a clearing schedule for a given graph requiring the minimum number of searchers. Let $S(t) = \{s_1(t), \ldots, s_K(t)\}$ (the locations of each searcher at time t), let $\mathbf{S} = S(0), \ldots, S(t_f)$ (the full search schedule), let $\overline{sn}(\mathbf{S}) = K$ (the number of searchers required for the schedule), and let $N_D(t|\mathbf{S})$ be the dirty set at time t given $\mathbf{S}$. Finally let $\Psi^F(t_f)$ be the set of feasible searcher paths on the search graph from time $t = 0, \ldots, t_f$. The optimization problem is to generate a feasible schedule $\mathbf{S}$ with minimal searchers K for which $N_D(t_f|\mathbf{S}) = \emptyset$ as shown in Equation 4.1. The variable t_f is left unspecified and can be modified as part of the optimization. Note that Equation 4.1 can (and likely does) have many possible optimal solutions.

$$\mathbf{S}^* = \underset{t_f, \mathbf{S}}{\operatorname{argmin}} \ \overline{sn}(\mathbf{S}) \ s.t. \ N_D(t_f|\mathbf{S}) = \emptyset \tag{4.1}$$

Equation 4.1 does not directly consider minimizing the length of the schedule t_f (i.e., the *clearing time* of the schedule). However, if many schedules are generated with a minimal number of searchers, the shortest schedule can easily be chosen. In addition, Chapter 5 shows that a variation of GSST can be used to minimize clearing time when given more searchers than the minimum number required.

This thesis examines search problems on graphs with the understanding that both the target and searchers exist on the vertices of the graph. This problem is referred to as node search, which is different from the edge search problem discussed in the literature (i.e., the target exists in the edges of the graph) (Parsons, 1976). Searching built environments lends itself to the node search formulation because rooms and hallways can be easily decomposed into nodes on the graph. The edge search formulation, on the other hand, describes a situation in which the edges on the graph are contaminated (e.g., with poison gas) or the search is performed in a system of tunnels. Kehagias et al. (2009a) showed that any edge clearing schedule is also a node clearing schedule, which allows algorithms from the edge search literature to be used to find node clearing schedules.

The proposed algorithm considers search schedules that are internal, monotone, connected, and rooted (defined below). We will refer to a *minimal* search schedule as one that requires the smallest possible number of searchers while maintaining these four characteristics. The *attained minimal* search schedule is the best schedule found thus far.

Definition A node search schedule $\mathbf{S}$ is internal, monotone, connected (IMC), and rooted if the following hold.

1. *Internal*: once placed on the graph, searchers can only move along the edges, and searchers are never removed from the graph.

2. *Monotone*: for all t, $N_C(t-1) \subseteq N_C(t)$.
3. *Connected*: for all t, $(N_C(t), E_C(t))$ is a connected subgraph of G.
4. *Rooted*: searchers can be placed only onto a single, pre-specified node called the *root* of the search.

Internal search schedules restrict the movement of the searchers to those feasible on the graph (i.e., searchers cannot teleport), and connected search schedules always maintain a connected subgraph of cleared nodes. Both of these characteristics are desirable in robotic search applications. The movements of robots in the real world are restricted to those that are physically possible. In addition, robot teams will often start in the same location, allowing the cleared regions to grow as a connected subgraph from that location.

4.2 Uninformed Implicit Coordination

The multi-robot guaranteed search problem is difficult to solve due to the tight coupling of the searchers in the joint path space and the necessity to search a large number of paths to find an optimal schedule. If an environment graph has a branching factor b and requires K searchers t_f steps to clear, the number of possible paths is $O(b^{Kt_f})$. As the environment size increases, both the number of searchers required and the time to clear will increase. This quickly leads to the infeasibility of any exhaustive search of the joint space.

In the previous chapter, FHPE+SA was presented, an approximation algorithm for solving the efficient search path planning problem. In the efficient search domain, searchers must maximize the probability of finding a non-adversarial target. FHPE+SA has searchers plan paths by enumerating all possible schedules to a finite-horizon. The robots do so greedily and sequentially, which allows for linear scalability in the number of searchers. Planning sequentially and sharing paths is a form of implicit coordination because the robots do not explicitly plan for their teammates.

FHPE+SA can be extended to guaranteed search with a simple modification. During finite-horizon planning, the searchers can limit their paths to those that do not cause recontamination (i.e., paths that do not allow clean nodes to become dirty). If the searchers plan sequentially and share their paths to construct a partial schedule incrementally, this leads to a piggyback effect during which each successive searcher extends the clearing schedule further in the environment. By iteratively planning on the receding horizon, the searchers can find a clearing schedule in this manner. We will refer to the resulting algorithm as Receding Horizon Greedy Clearing (RHGC).

Algorithm 4.1 gives a description of RHGC. The parameter d represents the horizon length in the receding-horizon planner. The running time of RHGC scales linearly with the number of searchers K, but exponentially with increasing d: $O(Kb^d)$, where

Algorithm 4.1 Receding Horizon Greedy Clearing (RHGC) for Guaranteed Search

```
 1: Input: Graph $G = (N, E)$, Start node $b \in N$
 2: $K \leftarrow 1$, $t \leftarrow 0$, $s_1(0) \leftarrow b$
 3: while $N_D \neq \emptyset$ do
 4:    for searcher $k = 1$ to $K$ do
 5:       % $P$ is a feasible path for searcher $k$ in the search graph to horizon $d$
 6:       % $\mathbf{S} = s_1(0), \ldots, s_K(0), \ldots, s_1(t+d), \ldots, s_{k-1}(t+d)$ is a partial schedule
 7:       % $N_D(t+d|\mathbf{S})$ is the dirty set at time $t+d$ given $\mathbf{S}$
 8:       $s_k(t+1), \ldots, s_k(t+d) \leftarrow \text{argmin}_P \, |N_D(t+d|P, \mathbf{S})|$
 9:    end for
10:    $t \leftarrow t+1$
11:    if no searcher can move without recontamination then
12:       % Add a new searcher and reset to start
13:       $K \leftarrow K+1$, $t \leftarrow 0$, $s_k(0) \leftarrow b$ for all searchers
14:    end if
15: end while
16: Output: Search schedule $\mathbf{S}$, clearing time $t$
```

b is the branching factor of the search graph. Thus, d should be set to a fairly small value, particularly if the branching factor is large.

RHGC is an application of linearly scalable implicit coordination to the guaranteed search domain. Unfortunately, uninformed implicit coordination can perform poorly because it ignores the requirement for tight coordination. In other words, the searchers sometimes must work together to make any progress in clearing the environment. Figure 4.2 gives an example where uniformed implicit coordination leads to a large number of searchers. The performance of RHGC is also compared to that of GSST in Section 4.4.

Given the potentially poor performance of implicit coordination during guaranteed search, it may be tempting to utilize market-based techniques or other methods of injecting explicit coordination into the search schedule. Such techniques would increase the coordination between the searchers, but at the cost of additional overhead in both communication and computation. GSST provides an alternative method for improving the performance of implicit coordination by exploiting the relatively easy problem of finding a guaranteed search schedule on trees.

4.3 The GSST Algorithm

This section presents GSST, an anytime algorithm for generating guaranteed node search schedules on graphs by utilizing their spanning trees. The algorithm takes advantage of the minimal edge search on the underlying spanning tree to guide node search on the full graph. A number of traversal variants are presented that utilize the utilize the underlying spanning tree in different ways. Though GSST does not have a performance guarantee with respect to the optimal solution, several traversal variants

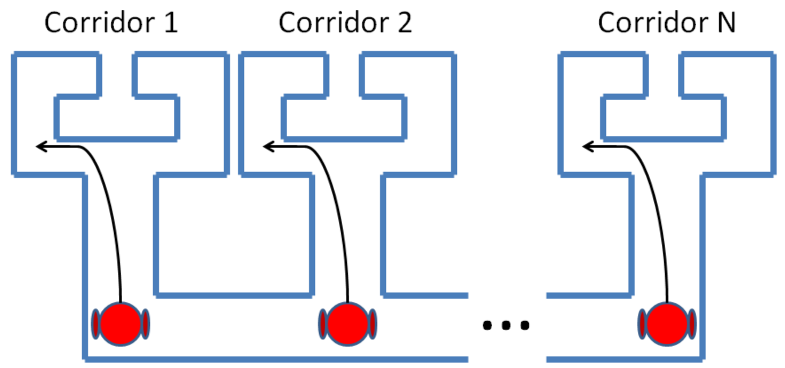


Figure 4.2: Example of an environment in which uninformed implicit coordination leads to a large number of searchers. If each searcher maximizes its own path on the finite-horizon, each will choose to go down a different corridor, requiring $N + 1$ searchers to clear the environment (i.e., the final searcher goes to assist each of the other N searchers with clearing their respective corridors). A minimal clearing schedule requires only three searchers regardless of the number of corridors: one searcher guards the main hallway, and two searchers clear the corridors together. In contrast, if the environment is discretized into a tree, a three-searcher solution can be found easily with Algorithm 4.3.

are shown to be probabilistically complete (i.e., they have a nonzero probability at each iteration of producing a search schedule using the minimal number of searchers).

4.3.1 Guaranteed Search on Trees

Minimizing the number of searchers to clear an arbitrary graph is an NP-hard problem, but linear-time algorithms exist for trees. This section describes a non-recursive version of the algorithm from Barrière et al. (2002). The basic idea is to start at the leaves of the tree and recursively join together subtrees, using the intuition that the search number must be increased when two or more subtrees being joined each require the same (maximal) number of searchers. In these cases, a guard needs to be left behind while one of the subtrees is cleared. This algorithm generates a minimal internal, monotone, connected schedule for a given tree in time linear in the nodes in the search graph. Eliminating the need for recursion allows for the direct application of implicit coordination (see Section 4.4).

It is assumed that the starting node of the searchers is known and the same for all searchers.[2] The starting location is denoted as $b \in N$. First, the edges on the tree $T = (N, E)$ are labeled with $\lambda : E \rightarrow Z^+$ as in Algorithm 4.2. The mapping $\lambda(e)$ describes the number of searchers that must move down the tree along that

[2]The rooted starting node assumption can be relaxed in many cases. A labeling can be generated using any starting location as a root, and the searchers can run the schedule from their current locations. However, this approach does not guarantee a minimal search schedule on the underlying tree.

Algorithm 4.2 Edge labeling for trees

```
 1: Input: Tree $T = (N, E)$, Start node $b \in N$
 2: $O \leftarrow$ leaves of $T$
 3: while $O \neq \emptyset$ do
 4:     $l \leftarrow$ any node in $O$, $O \leftarrow O \setminus l$
 5:     if $l$ is a leaf then
 6:         $e \leftarrow$ only edge of $l$, $\lambda(e) = 1$
 7:     else if $l$ has exactly one unlabeled edge then
 8:         $e \leftarrow$ unlabeled edge of $l$
 9:         $e_1, \ldots, e_d \leftarrow$ labeled edges of $l$
10:         $\lambda_m \leftarrow \max\{\lambda(e_1), \ldots, \lambda(e_d)\}$
11:         if multiple edges of $l$ have label $\lambda_m$ then
12:             $\lambda(e) \leftarrow \lambda_m + 1$
13:         else
14:             $\lambda(e) \leftarrow \lambda_m$
15:         end if
16:     end if
17:     if $l \neq b$ and $parent(l)$ has exactly one unlabeled edge then
18:         $O \leftarrow O \cup parent(l)$
19:     end if
20: end while
21: Output: Edge labeling $\lambda(E)$
```

edge during the search schedule. Next the edges are made directional by pointing them down the tree from the start node to the leaves. The edges are then doubled, and these new edges are given the opposite direction. The doubled edges are labeled with $\lambda(e_2) = -\lambda(e)$, where e_2 is the double of edge e. The negative values represent recursive steps back up the tree after clearing. The set of edges and their doubles are referred to as E'.

A minimal edge search schedule can be generated from this labeling in linear-time. Algorithm 4.3 is a non-recursive variation of the algorithm in prior work, and it will clear the edges of a tree T rooted at node b with the minimum number of searchers (Barrière et al., 2002). As described above, subtrees are joined together to generate a labeling, which encodes the number of searchers that must enter that subtree to clear it. Recursion is eliminated by decrementing the edge labels whenever a searcher traverses one. Thus, the edge labels represent the number of remaining searchers that must enter the subtree to clear it. The number of searchers necessary is equivalent to the edge labeling of an edge entering the start node, which is referred to as μ. Since this schedule is an edge clearing schedule, it is also a node clearing schedule of the underlying tree (Kehagias et al., 2009a).

4.3.2 Generating Spanning Trees

The algorithm described above for trees does not apply to arbitrary graphs because the edge labeling is not possible with cycles. However, additional searchers can be

Algorithm 4.3 Guaranteed search schedule for trees

```
1: Input: Tree $T = (N, E')$, Edge labeling $\lambda : E' \to Z$, Start node $b \in N$
2: % Define $s_k(t)$ as the node occupied by the $k \leq \mu$ searcher at time $t$
3: Calculate required searchers, $\mu$ = edge label into $b$
4: $t \leftarrow 0$, $K \leftarrow \mu$, $s_k(0) \leftarrow b$ for all $k$, $N_D \leftarrow N \setminus b$
5: **while** $N_D \neq \emptyset$ **do**
6:     $t \leftarrow t + 1$
7:     **for** all searchers $k$ **do**
8:         **if** searcher cannot move without recontamination **then**
9:             $s_k(t) \leftarrow s_k(t-1)$
10:        **else if** positive edge label exists incident to $s_k(t-1)$ **then**
11:            $s_k(t) \leftarrow$ node reached through *lowest* incident labeled edge
12:            Decrement $\lambda(e)$ of edge traversed
13:        **else**
14:            $s_k(t) \leftarrow$ node reached through incident negative labeled edge
15:            Increment $\lambda(e)$ of edge traversed
16:        **end if**
17:        $N_D \leftarrow N_D \setminus s_k(t)$
18:    **end for**
19: **end while**
20: Output: Search schedule $\mathbf{S}$, clearing time $t$
```

used as guards to transform an arbitrary graph $G = (N, E)$ into a tree $T = (N, S)$. The non-guard searchers can then traverse the resulting tree using the algorithm described above. This approach reduces the guaranteed search problem to that of generating a "good" spanning tree on which to base the search.

The problem of uniformly sampling the space of spanning trees has been heavily studied. Wilson's algorithm, based on the use of loop-erased random walks, is both efficient and conceptually simple (Wilson, 1996). The basic idea is to maintain a current tree, which consists initially of just the root. While vertices remain that are not in the tree, the algorithm performs a random walk from one such vertex, erasing cycles as they are created, until the walk encounters the current tree. The resulting distribution of spanning trees is uniform over all possible trees.

As an alternative to uniform selection, Algorithm 4.4 uses a randomized depth-first search to find a spanning tree. A depth-first search is started at the root, and an edge is chosen that leads to a node not in the tree. The parent of the new node is set to the previous node, and all incident edges to the new node are considered for random selection. As the search progresses, edges that would create cycles are marked as non-tree edges. If a node is encountered such that choosing any incident edges would create a cycle, the algorithm recurses and continues to randomly select edges of parents along the tree. This algorithm will not generate all possible spanning trees, but it focuses on trees that have non-tree edges incident to few nodes, which intuitively leads to trees that require fewer guards.

Algorithm 4.4 Randomized depth-first spanning tree generation

```
1: Input: Graph $G = (N, E)$, Start node $b \in N$
2: $V \leftarrow \emptyset$, $S \leftarrow \emptyset$, $B \leftarrow \emptyset$, $x \leftarrow b$
3: while some edges are in neither $S$ nor $B$ do
4:     $V \leftarrow V \cup x$, $R \leftarrow \emptyset$
5:     for all nodes $y$ adjacent to node $x$ do
6:         if $y \in V$ then
7:             $B \leftarrow B \cup e(x, y)$
8:         else if $y$ not already visited from $x$ then
9:             $R \leftarrow R \cup y$
10:        end if
11:    end for
12:    if $R$ is empty then
13:        $x \leftarrow$ parent of $x$
14:    else
15:        Choose random node $z \in R$
16:        Set parent of $z$ to $x$
17:        $S \leftarrow S \cup e(x, z)$, $x \leftarrow z$
18:    end if
19: end while
20: Output: Set of tree edges $S$, Set of non-tree edges $B$
```

4.3.3 Labeled Traversal

Given a set of tree edges S and a set of non-tree edges B, a naive clearing strategy can be found by assigning guards to a node incident to each non-tree edge and then searching the tree as in Algorithm 4.3. This technique ignores two important characteristics of the problem. First, adding a guard for every non-tree edge will likely be redundant. If several non-tree edges are incident to a single node, one guard will suffice for all of them. Second, the search schedule occurs over a time interval. Guards that are necessary at earlier times may be free to guard other edges at later times.

Algorithm 4.5 shows how these observations can be taken into consideration during search to generate a traversal strategy. We will refer to the resulting traversal as labeled traversal (GSST-L). The basic idea is to utilize the labeling of the spanning tree to generate moves for some searchers and to use additional searchers as guards when necessary to prevent recontamination. When a searcher can not perform the next move designated by the labeling, a guard is assigned to allow the move to occur. The algorithm progresses towards clearing the graph in this fashion. Algorithm 4.5 also demonstrates the anytime nature of GSST. As more spanning trees are generated, the lowest attained search number is stored. More schedules are generated with increasing computation, which leads to lower attained search numbers.

Note that Algorithm 4.5 can be modified for a more conservative use of guards. Instead of calling for a guard whenever a tree searcher reaches a node with incident non-tree edges, the algorithm can wait for all tree searchers to reach such nodes. This ensures that moving tree searchers will not release a previously stuck searcher.

Algorithm 4.5 Guaranteed Search with Labeled Traversal

```
1: Input: Graph $G = (N, E)$, Start node $b$
2: while time is available do
3:    Find spanning tree $T = (N, S)$
4:    Label edges $\lambda(S)$ using Algorithm 4.2
5:    Calculate required searchers, $\mu = \lambda(e)$, where $e$ is an edge entering $b$
6:    Generate $K_T \leftarrow \mu$ tree searchers
7:    while $G$ not cleared do
8:       Move tree searchers according to Algorithm 4.3
9:       if a tree searcher reaches a node $c$ with incident non-tree edge then
10:         if guard $k$ can move without recontamination then
11:            Move guard $k$ to node $c$
12:         else
13:            Generate new guard and move to node $c$
14:            $K_G \leftarrow K_G + 1$
15:         end if
16:      end if
17:   end while
18:   Record $\eta = K_T + K_G$
19: end while
20: Output: Search schedule $\mathbf{S}^*$ with lowest $\eta$
```

Furthermore, the algorithm can reassign tree searchers that are no longer necessary. For instance, three searchers may be needed to clear an early portion of the graph, but the remaining subgraph may only require two. In this case, a tree searcher can be reassigned as a guard after it is no longer needed as a tree searcher. These reassignments may affect search number because they can provide extra guards later in the search schedule. These two extensions are used in Section 4.4.

The guard allocation step can be seen as an instance of implicit coordination. When a searcher needs a guard, as determined by the underlying labeling, that information is broadcast to the team, and the closest available guard takes the assignment. It is assumed that two guards will not choose to cover the same node at the same time (see Section 4.4.3). Thus, the searchers utilize the shared spanning tree representation to help determine their task assignments. The generation of a spanning tree is a shared preprocessing step, which occurs before implicit coordination. One disadvantage of this technique is that, since the guard assignments are chosen greedily, a guard can be required to traverse a much longer distance than would be required by the optimal assignment. This drawback is alleviated somewhat in Chapter 5 where a guardless traversal is introduced.

4.3.4 Randomized and Label-Weighted Traversal

Several traversal variants of GSST are now considered that utilize the labeling in different ways. Algorithm 4.6 shows a randomized traversal strategy (GSST-R). When performing the randomized strategy, the searchers randomly select nodes adjacent to

Algorithm 4.6 Randomized Traversal

```
1: Input: Graph $G = (N, E)$, Start node $b$
2: Generate a spanning tree $T = (N, S)$
3: $t \leftarrow 0$, $N_C(0) \leftarrow b$, $K \leftarrow 1$, $s_1(0) \leftarrow b$
4: **while** $G$ not cleared **do**
5:     Randomly select edge $e \in S$ adjacent to $N_C$
6:     **if** a searcher can traverse $e$ without recontamination **then**
7:         Move to and traverse $e$ with that searcher
8:         Update $N_C$
9:     **else**
10:        Generate new searcher $K \leftarrow K + 1$
11:        $s_K \leftarrow b$
12:    **end if**
13: **end while**
14: Output: search schedule $\mathbf{S}$
```

the cleared set and move towards them. Unlike labeled traversal, randomized traversal does not utilize information from optimal search on the spanning tree. Randomized traversal is introduced to determine the effectiveness of the labeling on the quality of the search schedule (see Section 4.4).

A simple modification of random traversal is to weight the edge selection based on their labeling. This variation will be referred to as label-weighted traversal (GSST-LW). GSST-R and GSST-LW provide more variation in schedules than labeled traversal, and they are probabilistically complete in the number of searchers (see Theorem 4.1).

4.3.5 Label-Dominated Traversal

Another alternative is to traverse labeled edges that lead to branches with known clearing numbers. This can be done by labeling edges that lead to parts of the graph that are trees (subtrees of the graph). A list of searchers that can move without recontamination can be maintained during search. If an edge adjacent to N_C leads to a subtree of the graph, and enough free searchers are available, clearing this subtree can only improve the search schedule. The additional clearing moves can be determined randomly or using a label-weighted approach. This technique will be referred to as label-dominated traversal (GSST-LD). This traversal variant is introduced to combine the benefits of labeled traversal with those of randomized traversal (i.e., completeness and solution variation).

4.3.6 Correctness

For every graph G and spanning tree T, all presented variations of GSST generate a node clearing schedule with a finite number of searchers. Any spanning tree T of G contains all nodes in G. Also, any edge clearing schedule of T is also a node clearing schedule of T (Kehagias et al., 2009a). Thus, GSST-L will generate a clearing schedule of G if sufficient guards are added to prevent recontamination. Since the number of guards cannot be larger than the number of non-tree edges on the graph, the schedule uses finitely many searchers. GSST-R, GSST-LW, and GSST-LD start a single searcher at the root and progressively makes clearing moves with searchers that can move without recontamination. Searchers are added when non-contaminating clearing moves are impossible. Thus, progress is made at each step, and recontamination does not occur, which leads to a node clearing schedule.

All variations of GSST are guaranteed to generate a node clearing schedule with a finite number of searchers given a spanning tree. This analysis shows that every iteration of the anytime algorithm will generate a clearing schedule with a finite number of searchers.

4.3.7 Completeness

Theorem 4.1 shows that random and label-weighted traversal have a nonzero chance to find a minimal monotone, connected node clearing schedule on any graph. In other words, every iteration of the anytime algorithm has a chance to generate a minimal search schedule on the graph. Thus, if the algorithm were run for a sufficiently long time, it would find a minimal schedule on any graph.

Theorem 4.1 *At each iteration, both random traversal (GSST-R) and label-weighted random traversal (GSST-LW) of a uniformly generated spanning tree have a nonzero chance of yielding a minimal monotone/connected node search schedule on any graph.*

The intuition behind Theorem 4.1 is that the most important moves (w.r.t. number of searchers required) in a clearing schedule are the moves that clear a new node. The moves that occur within the cleared set cannot modify the number of searchers required. In an internal, connected, monotone schedule, a clearing move is always a move from an already cleared node to a dirty node. If we think of the clearing moves as edges on the graph, the resulting strategy describes a sequence of trees. The tree starts with a single node, the root node, and progresses until it encompasses each node in the graph when the entire environment is cleared. Since each clearing move extends the current tree by one node that is not in the tree, the resulting edges never create a cycle. Thus, it is reasonable to restrict the schedules to those that are described by an underlying spanning tree. Note, however, that this intuition only holds if the cleared set is connected and monotone as described in the problem formulation. The full proof of Theorem 4.1 is given in the Appendix.

It is conjectured that the completeness property also holds for label-dominated traversal, but a formal proof is not given. The completeness of labeled traversal is an open problem, which is discussed in Chapter 9. In addition, the proofs assume that a spanning tree generation strategy is employed that has a nonzero chance of producing any spanning tree. Uniform sampling has this property, but randomized depth-first sampling (Algorithm 4.4) does not. Thus, GSST may not be probabilistically complete when used in conjunction with depth-first sampling.

4.3.8 Solution Quality

This section gives theoretical analysis regarding the performance of GSST. This analysis is limited to labeled traversal. The performance of the search schedules generated by GSST is determined by the number of total searchers (guards plus tree searchers) required to clear the graph. Barrière et al. (2002) show that the worst-case bound on trees is $\mu \leq \log_2(|N|)$, which is essentially because a tree with branches has depth $\log_2(|N|)$. This is an upper bound on the number of tree searchers needed for a graph with $|N|$ nodes.

Labeled traversal requires both tree searchers and guards. As described above, the maximum number of guards needed is $|B| = |E| - |N| + 1$. Thus, the worst-case total number of searchers η for the algorithm is $\eta \leq \log_2(|N|) + |E| - |N| + 1$. It can now be shown that labeled traversal (Algorithm 4.5) will always terminate with a successful clearing schedule with at most η searchers (see Theorem 4.2).

Theorem 4.2 *Labeled traversal is guaranteed to terminate on an arbitrary graph $G = (N, E)$ with a successful clearing schedule with at most $\eta \leq \log_2(|N|) + |E| - |N| + 1$ searchers.*

Proof Let T be the spanning tree generated for G. This generates exactly $|E| - |N| + 1$ non-tree edges in G. The spanning tree requires at most $\log_2(|N|)$ tree searchers to clear (Barrière et al., 2002). If labeled traversal must generate a new guard whenever a guard is needed for a non-tree edge, it must generate $|E| - |N| + 1$ guards. This guards all non-tree edges allowing the spanning tree to be cleared. ∎

The maximum number of searchers in Theorem 4.2 is a worst-case bound on the performance of the first spanning tree generated for a graph. Randomly searching over the space of spanning trees and using temporal aspects of the search to reuse guards significantly reduces this number. It may be possible to construct a traversal strategy such that this worst-case is bounded relative to optimal, but this is left as an avenue for future work.

This analysis highlights the anytime nature of GSST. The searchers begin the initial step of generating spanning trees. If an acceptable schedule has been found or time has run out, the best spanning tree is utilized to perform the search. At any time during tree generation, a clearing schedule is available, though perhaps one requiring a large number of searchers.

4.3.9 Computational Complexity

Each iteration of GSST performs the following steps: (1) generate a spanning tree, (2) label the spanning tree, (3) determine the attained search number, and (4) generate the traversal. The first three steps can be performed in time linear in the number of nodes in the graph.

Finding a spanning tree using depth-first search requires visiting each node once to label its edges and is thus $O(|N|)$. The reader is directed to Wilson (1996) for a discussion of the complexity of uniform spanning tree generation. As described by Barrière et al. (2002), edge labeling and determining search schedules on trees can be done in linear-time. The non-recursive labeling (Algorithm 4.2) also is linear in the number of nodes, since it visits each node once.

Labeled tree traversal (Algorithm 4.3) with K searchers is $O(K|N|)$ since it requires an inner loop that determines the schedule for each individual searcher for $|N|$ clearing moves.[3] The worst-case number of searchers to clear a tree is $O(\log |N|)$ (see above), which leads to $O(|N| \log |N|)$ computation. This approach has the advantage that it allows the computation to be distributed among the searchers.

Replacing the inner loop with a centralized planner that moves multiple searchers during each iteration reduces the computation to $O(|N|)$. This modification is equivalent to the recursive planner used by Barrière et al. (2002) on trees, except that it also counts guard placements. The randomized traversal variants (e.g., Algorithm 4.6) also are $O(|N|)$ because they must make $|N|$ random choices of frontier nodes to generate a clearing schedule.

The output of all planners described above may contain teleporting guard moves, which will require further processing to make the schedule internal. For labeled traversal of arbitrary graphs, the search schedules of guards can be determined using Dijkstra's algorithm between guard points, which is $O(|E| + |N| \log |N|)$ per guard. As described above, GSST may require up to $|E| - |N| + 1$ guards. A feasible guard schedule is always possible because the search schedule is monotone and connected. Note that this step only needs to be performed once when a schedule with sufficiently few searchers is found and ready to be executed.

In some cases, there may be a limited number, K_{max}, of available searchers, and schedules requiring more than K_{max} searchers may be considered infeasible. In these cases, the determination of the attained search number can be stopped when $K_{curr} > K_{max}$, and a new spanning tree can be generated at this time. Additionally, K_{max} can be set to the best solution so far, which will throw out any schedule as soon as it generates more than the current attained minimal. These modifications save computation and lead to finding a desirable schedule more quickly.

GSST is limited by the number of iterations (i.e., number of spanning trees) needed to find a schedule with sufficiently few searchers. For large graphs, an exponential number of spanning trees are possible. GSST leverages the fact that many spanning trees will yield good search schedules (though not necessarily minimal ones).

[3]Note that a clearing schedule on a tree may require up to $2|N|$ moves since the searchers may need to recurse up the tree.

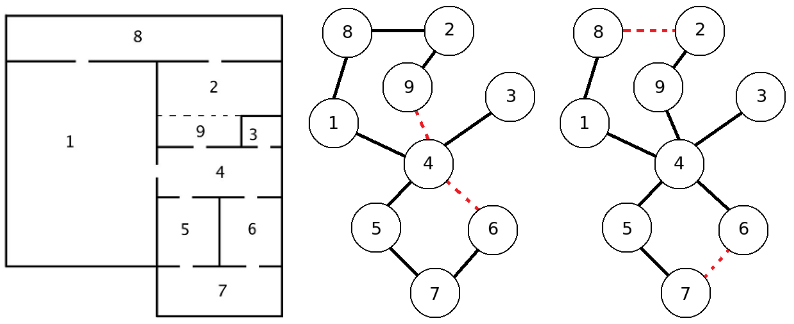


Figure 4.3: Example discretization of house environment (top) and two example spanning trees of resulting graph (bottom). Black edges (solid lines) are spanning tree edges, and red edges (dashed lines) are non-tree edges. If searchers start in cell 3, the left spanning tree gives a two searcher clearing schedule while the right spanning tree gives a three searcher clearing schedule.

4.3.10 Example

This section provides an example that demonstrates the importance of the choice of underlying spanning tree when using GSST. For explanatory purposes, labeled traversal is used for this example. Figure 4.3 shows a small house environment. If the spanning tree on the left of Figure 4.3 is found, then two tree searchers are required to clear the tree. Assume that the searchers start in cell three. The searchers first move down the spanning tree to cell four. Then one searcher must remain at cell four while the other searcher clears the rest of the graph. Since cell four is the only cell that needs to be guarded to remove the non-tree edges, this yields a two searcher clearing schedule of the original graph. Two is also the minimal number of searchers capable of clearing in this example.

Even in this simple example, all spanning trees do not yield a minimal (two searcher) schedule. Consider the right spanning tree in Figure 4.3. Ignoring the non-tree edges, this spanning tree requires two searchers to clear. However, when clearing the original graph, an extra guard must be placed on one of the cycles. The guard at cell four cannot be reused because its movement would cause recontamination. Thus, this spanning tree yields a three searcher schedule on the original graph due to the necessity of guarding the non-tree edges. These types of scenarios are the motivation for generating many spanning trees and choosing the best schedule.

4.4 Guaranteed Search Experiments

Experiments were conducted on a 3.2 GHz Pentium 4 processor with 2 GB RAM running Ubuntu Linux. Three methods for generating spanning trees were implemented and tested (see below).

1. Spanning tree enumeration (Char, 1968): Char's algorithm for enumerating all spanning trees. This is a brute force method for generating all possible spanning trees, which is computationally viable only on small graphs with few spanning trees.

2. Uniform sampling (Wilson, 1996): Wilson's algorithm for uniformly sampling the space of spanning trees using loop-erased random walks. The algorithm generates a random spanning tree sampled uniformly from the space of all spanning trees.

3. Depth-first sampling (Section 4.3): randomized depth-first search to generate spanning trees. This technique does not sample the entire space of trees, but it attempts to bias sampling towards trees requiring a small number of guards.

A summary of the five traversal variants described in Section 4.3 is given below.

1. GSST-L: labeled traversal. The labeling of the underlying spanning is used deterministically to guide the search (see Algorithm 4.5).

2. GSST-R: randomized traversal. The underlying spanning tree is traversed randomly. The labeling is not used.

3. GSST-LW: label-weighted traversal. The underlying spanning is traversed in a randomized manner with weights determined by the labeling.[4]

4. GSST-LD: label-dominated traversal. Subtrees of the graph are deterministically cleared as available and other moves are determined randomly.

5. GSST-LDW: label-dominated/weighted traversal. Subtrees of the graph are deterministically cleared as available, and other clearing moves are determined in a randomized manner with weights determined by the labeling.

4.4.1 Simulated Indoor Environments

With robotic search in mind, GSST was tested on the two complex indoor environments introduced in the previous chapter (Figure 3.2). The first map is a floorplan of the third floor of Newell-Simon Hall at Carnegie Mellon University. The second is a map of the first floor of the National Gallery of Art in Washington, DC. The Newell-Simon Hall ("office") environment has two major cycles, and the National Gallery ("museum") environment has many cycles by which the target can escape capture.

[4]The exact weighting function used is to step through the frontier edges and generate a random integer $\tau \in [0, \lambda_{max}]$, where λ_{max} is the maximum label of all frontier edges. A frontier edge e is accepted for traversal if its label $\lambda(e) \leq \tau$. Any probabilistic function that is inversely proportional to the weights could be used in place of this.

The office map is 100 m $\times$ 50 m discretized into 60 cells with 64 edges. Kirchhoff's matrix-tree theorem shows that the office has 3604 different spanning trees. The museum map is 150 m $\times$ 100 m discretized into 70 cells with 93 edges ($5.3 * 10^{14}$ spanning trees). Thus, the space of spanning trees can be exhaustively searched for the office but not for the museum. It is not immediately obvious, but the office can be cleared with three searchers and the museum with five.[5]

Tables 4.1 shows a comparison of different spanning tree generation techniques and traversal methods on the museum and office maps. GSST is able to generate 100,000 schedules in under a minute (often under 30 seconds) on these complex maps.[6] Also note that all traversal variants yield similar running times, with the exception of GSST-R that does not need to perform the labeling step. Traversal methods using spanning tree labeling (i.e., all but GSST-R) generate more minimal schedules. These results demonstrate the gain from using the optimal traversal of the underlying spanning tree to guide search. The gain from using the labeling is greater on the office map than on the museum map, which is expected because the office map has fewer cycles and is thus closer to a tree.

In the office and museum, depth-first sampling yields a higher proportion of minimal schedules due to its bias towards graphs requiring fewer guards. However, particularly in the office, depth-first sampling severely limits the space of spanning trees, which leads to only ten distinct schedules (two of which are minimal). This demonstrates the incompleteness of depth-first sampling along with its tendency to produce minimal search schedules on the reduced space of trees.

Figure 4.4 illustrates the anytime behavior of GSST by showing the attained search number with increasing numbers of generated spanning trees. As above, in both environments, depth-first sampling generates solutions with fewer searchers much more quickly than uniform sampling. In addition, Figure 4.5 displays histograms of the number of searchers to clear the graphs for many different spanning trees. Depth-first sampling clearly generates trees that lead to clearing schedules with fewer searchers in these environments. Figure 4.1 shows example spanning trees yielding minimal search schedules in both environments.

4.4.2 Performance Comparison

Here GSST is compared to several competing algorithms using three additional environments (shown in Figure 4.6). The hallway, cave, and Gates Hall environments were introduced by Gerkey et al. (2005) to test their PARISH algorithm. Their discretization is used for the Gates and hallway environments, and a constrained Delaunay triangulation was generated for the cave environment (see Section 4.1) since

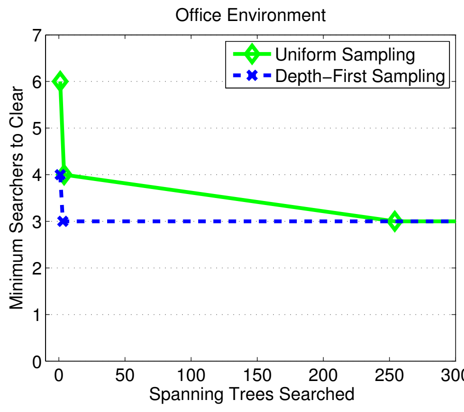
[5]While it is not proven that five is the minimal search schedule in the museum, a four searcher clearing schedule has not been found on this map using any method.

[6]Note that even though the office environment has only 3600 spanning trees, a much larger number of schedules are possible using the randomized traversal variants.

Table 4.1: Comparison of variations of GSST in two environments. The first column gives the method of edge traversal, the second column the minimum number of searchers attained, the third column the percentage of minimal solutions over all computed solutions, and the fourth column the total execution time (for 100,000 schedules). The same minimal schedule may be counted more than once in the percent minimum. The starting node is fixed throughout the trials. The traversal strategies that utilize the labeling of the underlying spanning tree yield schedules with fewer searchers. GSST-LD and GSST-LDW provide the best schedules in most cases by combining the labeling with randomization of the schedule.

Office	**Uniform ST gen.**		
Traversal	Min	% min	Time
GSST-L	3	0.7780%	20.1886
GSST-R	3	0.0040%	15.4610
GSST-LW	3	0.0320%	20.7670
GSST-LD	3	0.3540%	21.5415
GSST-LDW	3	0.4070%	22.1352
	DFS ST gen.		
GSST-L	3	19.9880%	18.1599
GSST-R	3	0.0510%	13.8348
GSST-LW	3	0.5390%	20.1243
GSST-LD	3	20.0480%	20.4685
GSST-LDW	3	20.005%	21.0810

Museum	**Uniform ST gen.**		
Traversal	Min	% min	Time
GSST-L	5	0.0020%	47.5863
GSST-R	5	0.0010%	31.6156
GSST-LW	5	0.0040%	45.4039
GSST-LD	5	0.0090%	45.5225
GSST-LDW	5	0.0110%	45.9819
	DFS ST gen.		
GSST-L	5	0.4220%	47.1498
GSST-R	5	0.3360%	29.0825
GSST-LW	5	0.5400%	48.0258
GSST-LD	5	0.8650%	47.4625
GSST-LDW	5	0.8330%	48.0719



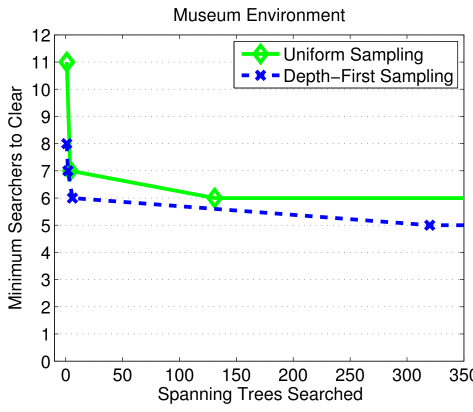


Figure 4.4: Demonstration of anytime behavior of GSST in the office (top) and museum (bottom). In both environments, depth-first spanning tree generation more quickly generates a spanning tree with few searchers. Labeled traversal is used in these experiments.

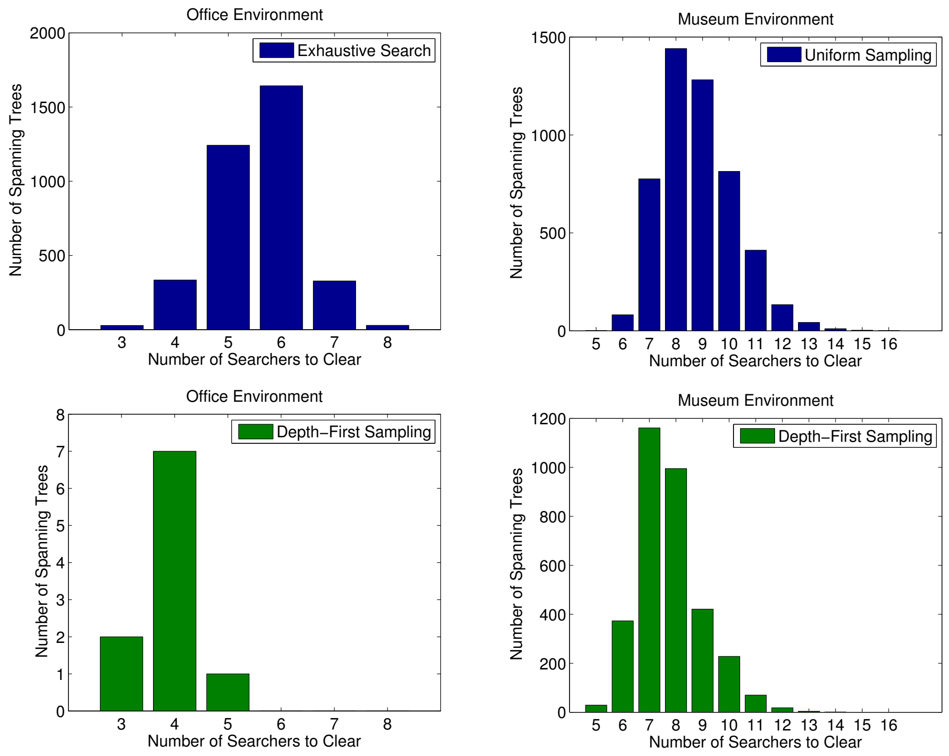


Figure 4.5: Histograms of number of searchers to clear the office (left) and museum (right) using labeled traversal on many different spanning trees. In the office, exhaustive search is compared to depth-first sampling. Depth-first sampling limits the space of spanning trees searched but generates trees that require fewer searches. In the museum, uniform sampling of 30,000 trees is substituted for exhaustive search. Again, depth-first sampling generates spanning trees requiring fewer searchers.

the discretization by Gerkey et al. is not available.

Table 4.2 shows the attained search number (i.e., the minimal number of searchers with which a clearing schedule was found) for GSST, PARISH, RHGC, and the iterative greedy algorithm. PARISH is a stochastic hill-climbing approach that has much in common with market-based techniques (Gerkey et al., 2005).[7] The use of receding-horizon planning with sequential allocation (RHGC) for guaranteed search is discussed in Section 4.2. The iterative greedy algorithm uses an approach similar to dynamic programming to generate clearing schedules (Kehagias et al., 2009b). The version of GSST employed in this section uses uniform spanning tree generation and labeled traversal. GSST generated 10,000 spanning trees before returning the best solution. This took less than 10 seconds in the larger environments. RHGC and

[7]The results reported in Table 4.2 for PARISH are taken directly from (Gerkey et al., 2005) and that article's supplementary videos (Gerkey, 2004) in which some of the same environments are examined. See Chapter 5 for a more extensive quantitative comparison between an improved version of PARISH and a variant of GSST.

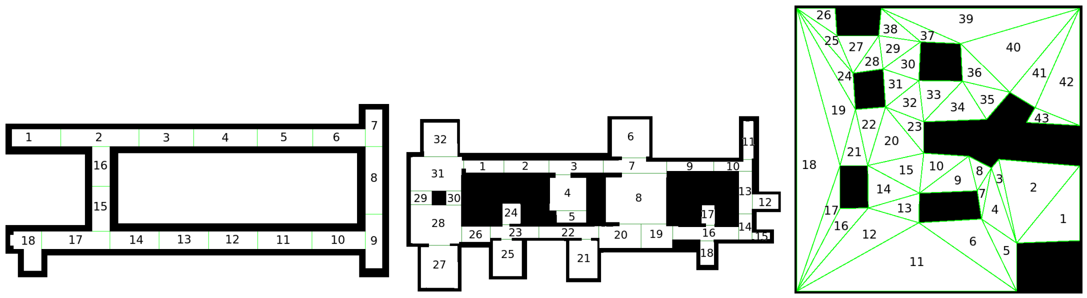


Figure 4.6: Additional simulated environments used to compare GSST to competing algorithms. The hallway (top) and Gates (middle) environments use the same discretizations as (Gerkey et al., 2005). The cave (bottom) was discretized using a constrained Delaunay triangulation. Green lines denote cell boundaries.

Table 4.2: Comparison of attained search numbers of GSST (Algorithm 4.5) with RHGC (Algorithm 4.1), PARISH, and Iterative Greedy. GSST and RHGC were both developed in this thesis. The Iterative Greedy results were taken from Kehagias et al. (2009b), and the PARISH results were taken from Gerkey et al. (2005), where the office and museum were not considered. GSST yields the lowest attained search number in all environments, and the improvement is higher the more complex the environment.

	Hallway	Cave	Gates	Office	Museum
PARISH	2	4	5	-	-
RHGC	2	3	4	6	7
Iter. Greedy	2	3	4	4	6
GSST	2	3	3	3	5

iterative greedy were run with an increasing number of searchers (e.g., $K = 1, 2, \ldots$) until a clearing schedule was found. The horizon depth for RHGC was set to $d = 6$.

GSST yields the lowest attained search numbers in all environments versus these competing algorithms. The improvement in attained search number increases with the complexity of the environment (i.e., number of nodes and edges), which demonstrates the effectiveness of utilizing the underlying spanning tree to guide implicit coordination on large graphs. In addition, the linear scalability and anytime capabilities of GSST allow it to remain effective in large environments like the office and museum maps.[8]

[8]Multimedia extensions (see Appendix A) show playback of clearing schedules using GSST on several test maps.

4.4.3 Human-Robot Teams

The experiments above were run on a single processor with shared memory, and they do not demonstrate GSST's potential to be distributed across a team of searchers. In addition, the schedules in the simulated experiments were generated offline, and the paths of the robots were assumed to be completely synchronized. In real-world applications, it is possible to make some modifications to the schedules generated by GSST online. If a searcher is behind in its schedule, then other searchers can continue their schedules until they reach a point where moving would cause recontamination. Furthermore, if an entire branch of the tree is cleared fortuitously (e.g., by some uncontrolled searchers or information), the schedule can prune this branch and continue without clearing it. Note, however, that some online modification could modify the attained search number of the schedule (e.g., a searcher moving to clear an area instead of moving to a required guard position), and these should be avoided.[9]

To test the distributed and online capabilities of GSST, experiments were conducted with a human/robot search team on a single floor of an office building (shown in Figures 4.7 and 4.8). Two humans and a single Pioneer robot share their position information across a wireless network, and the entire team is guided by a decentralized implementation of labeled traversal. The robot's position is determined by laser Adaptive Monte-Carlo Localization (Thrun et al., 2005), and it is given waypoints through the Player/Stage software (Gerkey et al., 2003). The robot uses the built in Wavefront planner and Vector Field Histogram (VFH) obstacle avoidance to follow the waypoints specified by the GSST module. The robot uses a 180 degree FOV camera with a video feed to the humans as its clearing sensor. The humans input their position through the keyboard, and they are given waypoints through a GUI. Figure 4.9 shows a diagram describing the decentralized system architecture. The searchers communicate through the GSST modules, which receive position feedback and provide waypoints back to the team. This architecture does not require a centralized control mechanism. In addition, the GSST modules are anonymous: they do not need to know whether the position/clearing input comes from a human or a robot.

Prior to search, processors on each searcher randomly generated spanning trees in parallel until a three-searcher schedule was found. Each searcher continuously shared its three-searcher spanning tree that generated the lowest estimated clearing time. If a spanning tree with a lower clearing time was found by one searcher, it was propagated through the network. For small environments, it would be easy to partition the space of trees to search all of them systematically, but such a technique was not necessary to find a good schedule. After one second of spanning tree generation, execution was performed as described below using the decentralized network.

After the spanning tree generation step, each searcher sequentially checked the

[9]A method for extending a variation of GSST to situations requiring dynamic replanning is discussed in Chapter 5.

Figure 4.7: Volunteer firefighter searching with a Pioneer mobile robot. The robot and humans execute a schedule that clears the environment of any worst-case adversarial target. The decentralized GSST algorithm allows the robot and humans to fill the necessary search roles. The agents share their paths and a spanning tree that guides the search.

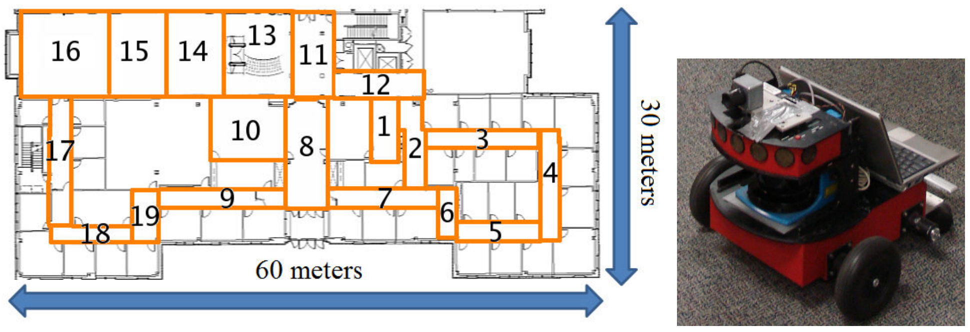


Figure 4.8: Map of office building (top) used for experiments with a human/robot search team. Pioneer robot searcher (bottom) with laser rangefinder and camera. The robot used the laser scanner for localization, and it used the camera to provide a video feed of the search environment. The robot and human team started their search at the building entrance in cell 8.

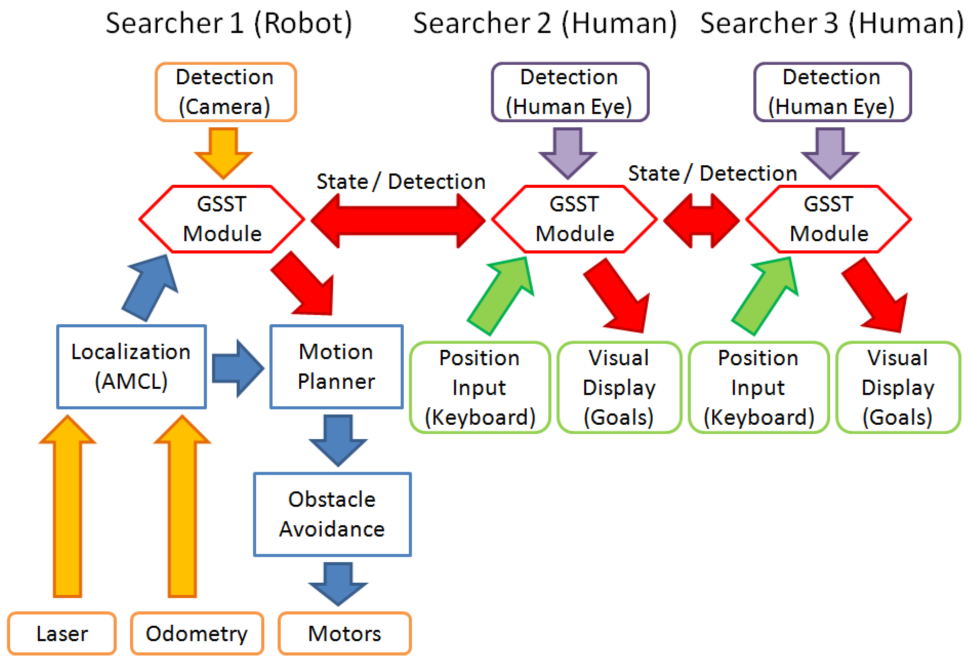


Figure 4.9: Diagram describing system architecture for human-robot search experiments. The decentralized GSST modules receive position feedback from the heterogeneous search team and then provide search waypoints back to the team.

spanning tree labeling to determine it next move. If the move was possible without recontamination, based on the positions of the other searchers, the move was taken, the corresponding label decremented, and the new label shared with the team. If the move would cause recontamination, the searcher waited and broadcast its state to the rest of the team. As other searchers replanned, they would see that a guard was needed, and move to the necessary location to allow the clearing schedule to continue. Each searcher repeated this planning step whenever reaching a cell centroid. Thus, the search was modified on-the-fly as necessary during clearing. For instance, one of the human searchers performed its clearing schedule more slowly than expected. The other searchers were able to continue their schedules without that human until they reached a point of recontamination. This implementation avoided strict synchronization, which would have required more costly communication.

It is important to note that incorrect moves could be made if information about labeling or positions were delayed. For instance, if an edge labeling was decremented because a searcher had traversed it, later searchers reaching the incident node would need to know that information to plan their next move. The proposed method has the advantage that discrete events are all that need to be shared. A searcher must share information about when it reaches a cell and which cell it is going to next. In this experiment, the network delay was not significant enough to cause a problem. For a large network delay, it might be necessary to invalidate moves made with a lack of information once the information becomes available. As long as recontamination was not caused, the schedule could be repaired. Thus, the primary concern in scenarios with high network delay would be to avoid recontamination whenever possible.

The first three-searcher schedule was generated very quickly (less than a second), and the network then decided on the schedule with the lowest clearing time as described above. The human-robot search team proceeded to clear the floor of a potential adversary. The schedule clears the left portion of the map first and then continues to clear the right portion. In addition to showing the distributed and online capabilities of GSST, the results with a human/robot team demonstrate the feasibility of the communication and computational requirements on a small team.

4.5 Chapter Summary

This chapter presented GSST, an anytime guaranteed search algorithm applicable to complex physical environments. GSST generates guaranteed search schedules on arbitrary graphs by leveraging the easier problem of generating a schedule on an underlying spanning tree. The GSST algorithm works in an anytime manner by quickly generating an initial schedule with potentially many searchers and then producing more schedules with increasing computation. Several variations of GSST were presented that utilize the underlying spanning tree in different ways, and a novel method for generating random spanning trees was given that tends to yield search schedules with few searchers. Theoretical analysis shows that GSST produces a feasible clearing schedule at every iteration, and several variations of GSST are probabilistically complete. In addition, GSST is linearly scalable in the number of nodes in the search graph, making it applicable to very large environments. Loose performance bounds were also given based on the worst-case number of searchers required to traverse the underlying spanning tree.

The results demonstrate both the scalability and effectiveness of GSST. The current implementation generates over 2000 schedules per second in an environment with 70 nodes and 93 edges. The results show that GSST outperforms three competing algorithms in five environments ranging from a cave to a museum. In some cases, the required number of searchers is reduced by 50% over competing algorithms. In addition, the feasibility of GSST was demonstrated using an implementation on a human-robot search team. The distributed implementation utilizes existing wireless infrastructure to communicate the searchers' positions, and the team effectively clears a floor of an office building of any potentially adversarial target. Furthermore, the schedule is modified online to account for poor synchronization between the searcher paths. The implementation of a clearing schedule on this scale highlights the real-world applicability of GSST to robotic search.

GSST is an implicit coordination algorithm, which allows for each agent to plan for itself without considering the joint planning space. The agents share the spanning tree of the environment as an informed representation, and they communicate information about their actions to execute a joint schedule. This indirect form of coordination provides the necessary coupling to generate schedules in difficult clearing instances without requiring exponential computation. The application of implicit coordination in tightly coupled domains avoids the computational overhead of running market-based strategies or auctions. The computational simplicity and decentralized operation of implicit coordination are preserved, and high-performance is achieved through the use of the informed representation.

The version of GSST proposed in this chapter does not directly optimize for clearing time. Even though spanning trees with lower clearing times can be chosen, there is no mechanism for utilizing additional searchers beyond the minimum number. The use of implicit coordination to select guard placements can also lead to overly long clearing schedules. In many applications, clearing time is nearly as important as number of searchers used, and a fixed number of searchers may be given beforehand. In addition, using a purely adversarial assumption precludes the use of any knowledge of the target's movement beyond the worst-case. In the next chapter, it is shown that a variant of GSST can be combined with FHPE+SA to optimize clearing time and provide good performance with respect to a target motion model.

Chapter 5

Combining Efficient and Guaranteed Search

As before, imagine you are the leader of a team of agents (humans, robots, and/or virtual agents), and you enter a building looking for a person, moving object, or contaminant. You wish either to locate a target in the environment or authoritatively say that no target exists. In some special cases, you may have a perfect model of how the target is moving; however, in most cases you will only have an approximate model or even no model at all. To complicate the situation further, the target may be adversarial and actively avoiding being found. The algorithms discussed previously would force you, the leader, to make a choice in this situation. Do you make the worst-case assumption and choose to treat the target as adversarial? This would allow you to utilize graph search algorithms to guarantee finding the target (if one exists), but it would not allow you to take advantage of any model of the target's motion. As a result your search might take a very long time. Or do you decide to trust your motion model of the target and assume that the target is non-adversarial? This assumption would allow the use of efficient (average-case) search methods from the optimization literature, but it would eliminate any guarantees if the model is inaccurate. In this case, your target may avoid you entirely. It is necessary to make one of these choices because no existing method provides fast search times is the model is correct and also guarantees finding a target if the model is wrong.

The methods discussed in the previous chapters treat multi-agent search as either a worst-case problem (i.e., clear an environment of an adversarial evader with potentially infinite speed), or an average-case problem (i.e., minimize average capture time given a model of the target's motion). The current chapter proposes treating search as a resource allocation problem, which leads to a scalable implicitly coordinated algorithm for generating schedules that clear the environment of a worst-case adversarial target *and* have good average-case performance considering a non-adversarial motion model.

This chapter proposes a novel algorithm that augments a guaranteed clearing schedule with an efficient component based on a non-adversarial target motion model. GSST (see Chapter 4) is extended to optimize clearing time, and it is augmented with a decentralized finite-horizon planning method (see Chapter 3) in which agents implicitly coordinate by sharing plans. It is shown that the average-case performance of the combined algorithm is bounded, and it is demonstrated that this algorithm can be used in an anytime fashion by providing additional search schedules with increasing runtime. This approach produces a family of clearing schedules that can easily be selected before the search or as new information becomes available during the search. The resulting combined search algorithms utilizes implicit coordination and allows for decentralized and online operation. The approach is validated using extensive simulated experiments as well as on a human-robot search team. The contribution of this chapter is the first algorithm that provides guaranteed solutions to the clearing problem and, given additional knowledge of the target's behavior, improves average performance.[1]

5.1 Combined Search Problem Setup

In this section, the search problem is defined with respect to both a worst-case adversarial target and a non-adversarial target. The formal connection is shown between worst-case search and the guaranteed search problem discussed in Chapter 4. As before, assume we are given K searchers and a graph $G = (N, E)$ with $|N|$ nodes and $|E|$ edges. The nodes in the graph represent possible locations in the environment, and the edges represent connections between them. At all times $t = 1, 2, \ldots, T$, the searchers and the target exist on the nodes of this graph and can move through an edge to arrive at another node at time $t + 1$. Given a graph of possible locations and times, we can generate a time-evolving, time-unfolded graph $G' = (N', E')$, which gives a time-indexed representation of the nodes in the environment (see Chapter 3 for a formal definition).

The searchers' movements are controlled, and they are limited to feasible paths on G'. We will refer to the time-stamped nodes visited by a searcher k as $A_k \subseteq N'$, and the combined set of visited time-stamped nodes as $A = A_1 \cup \ldots \cup A_K$.[2] The searchers receive reward by moving onto the same node as the target. This reward is discounted by the time at which this occurs. Given that a target visits time-stamped nodes Y, the searchers receive reward $F_Y(A) = \gamma^{t_A}$, where $t_A = \min \{t : (u, t) \in A \cap Y\}$ (i.e., the first time at which Y intersects A), with the understanding that $\gamma \in (0, 1)$, $\min \emptyset = \infty$, and $\gamma^\infty = 0$. Thus, if their paths do not intersect the target, the searchers receive zero reward.

[1]The algorithm proposed in this chapter was originally introduced by Hollinger et al. (2010b).

[2]As in Chapter 3, this formulation assumes that multiple searchers can visit the same node at the same time.

This chapter considers two possible assumptions on the target's behavior, which yield the average-case and worst-case search problems respectively. If a non-adversarial assumption is made on the target's behavior, a target motion model independent of the locations of the searchers can be utilized. This yields a probability $\Pr(Y)$ for all possible target paths $Y \in \Psi$, where Ψ is the set of feasible paths the target can take. The optimization problem in Equation 5.1 can now be defined.

$$A^* = \operatorname*{argmax}_{A \subseteq N'} \sum_{Y \in \Psi} \Pr(Y) F_Y(A), \tag{5.1}$$

where $A = A_1 \cup \ldots \cup A_K$, and each A_k is a feasible path for searcher k. Equation 5.1 maximizes the *average-case* reward given a motion model defined by $\Pr(Y)$. Note that if the target's motion model obeys the Markov property, its location can be estimated efficiently at each time step using matrix algebra. This optimization problem was considered in detail in Chapter 3 (see Equation 3.2).

The average-case optimization problem in Equation 5.1 does not consider the possibility that the motion model may be incorrect. An alternative assumption on the target's behavior is that it actively avoids the searchers as best possible. For the search problem, this implies that the target chooses path Y that minimizes $F_Y(A)$, which yields the game theoretic optimization problem in Equation 5.2.

$$A^* = \operatorname*{argmax}_{A \subseteq N'} \min_{Y} F_Y(A), \tag{5.2}$$

where $A = A_1 \cup \ldots \cup A_K$, and each A_k is a feasible path for searcher k. Here, the searchers' goal is to maximize the *worst-case* reward if the target acts as best it can to reduce reward.[3] This optimization problem is related to the guaranteed search problem in Chapter 4 (see Equation 4.1), but rather than minimizing the number of searchers, it minimizes clearing time given a fixed number of searchers.

Given the worst-case and average-case assumptions on the target's behavior (either of which could be correct), the searchers' goal is to generate a feasible set of paths A such that the reward of both optimization problems are maximized. One option is to use scalarization to generate a final weighted optimization problem as shown in Equation 5.3. The variable α is a weighting value that can be tuned depending on how likely the target is to follow the average-case model.

$$A^* = \operatorname*{argmax}_{A \subseteq N'} \quad \alpha \sum_{Y \in \Psi} \Pr(Y) F_Y(A) + (1 - \alpha) \min_{Y} F_Y(A), \tag{5.3}$$

where $A = A_1 \cup \ldots \cup A_K$, each A_k is a feasible path for searcher k, and $\alpha \in [0, 1]$ is a weighting variable to be tuned based on the application.

While scalarization is a viable approach, it also makes the problem more difficult

[3]It should be noted that in the worst-case formulation, the searchers must reveal their entire paths before the target chooses its path; so both sides must have a deterministic optimal strategy. The searchers choose a schedule, and the target chooses the path that best evades that schedule.

to solve. Note that the underlying function $F_Y(A)$ is nondecreasing and submodular (see Chapter 3) as is its expectation in Equation 5.1. In fact, Equation 5.1 is the efficient search (MESPP) problem, which can be optimized using the FHPE+SA algorithm. FHPE+SA yields a performance guarantee on the finite-horizon and has been shown to perform near-optimally in practice. In contrast, $\min_Y F_Y(A)$ is nondecreasing but is *not* submodular. Thus, FHPE+SA does not provide a performance guarantee and, in fact, performs poorly in practice. Furthermore, it is shown below that the optimization problem in Equation 5.2 does not yield any bounded approximation (unless $P = NP$).[4] Thus, scalarization combines an easier problem with a more difficult problem, which prevents exploiting the structure of the easier MESPP problem.

This chapter proposes treating the combined search problem as a resource allocation problem. More precisely, some searchers make the average-case assumption, and others make the worst-case assumption. A common scenario where this approach can improve the search schedule is when a portion of the map must be cleared before progressing. In many cases, redundant searchers are assigned as guards and must wait for the other searchers to finish clearing. Assigning fewer guards to a different location can often free some searchers to explore the uncleared areas, while still preventing recontamination. An example of this case is shown in the human-robot experiments in Section 5.4.

If the searchers are properly allocated to the average-case and worst-case tasks, search schedules can be generated with good performance under both assumptions. The decentralized nature of the multi-robot search task makes this approach feasible by allowing different robots to optimize the two separate components of the combined problem. In addition, the average-case and worst-case searchers often fortuitously benefit from one another's actions. The question now becomes: how many searchers should be assigned to each task? Several observations relating worst-case search to graph theoretic node search can help answer this question.

The graph theoretic node search optimization problem is to find a feasible schedule $\mathbf{S}$ for a minimal number of searchers such that $\mathbf{S}$ is a clearing schedule. In other words, find a schedule that clears the environment of an adversarial evader using the fewest searchers. Most graph search algorithms, including GSST in Chapter 4, focus on minimizing the number of searchers, and they do not directly minimize clearing time. Further analysis will now show the connection between graph theoretic node search and the worst-case search problem described above.

Propositions 5.1 and 5.2 show the connection between node clearing and the worst-case search problem defined in Equation 5.2. Note that Propositions 5.1 and 5.2 hold for both monotone and non-monotone clearing schedules. In other words, the

[4]Note that the problem of minimizing the number of *searchers* to clear admits an approximation guarantee. However, the problem of minimizing the discounted *time* to clear does not. See also Borie et al. (2009) for a discussion of related complexity results for special case graphs.

propositions hold regardless of whether recontamination is allowed in the schedule.[5]

Proposition 5.1 *The value* $\min_Y F_Y(A)$ *is greater than* 0 *if and only if* A *is a clearing schedule (i.e., a set of paths that clear the environment of any target within it).*

Proof Assume that A is not a clearing schedule and $\min_Y F_Y(A) > 0$. Since A is not a clearing schedule, this implies that one or more nodes in G are dirty (i.e., may contain a target) at all times $t = 1, \ldots, T$. W.l.o.g. let Y be a target path that remains within the dirty set for all t. Such a path is feasible due to the assumptions of the evader and the recontamination rules. The dirty set at a given time is by definition not observed by the searchers at that time. Thus, $A \cap Y = \emptyset$, which implies that $F_Y(A) = 0$ for these A and Y. If the target chooses this path, we have a contradiction.

Now, assume that A is a clearing schedule and $\min_Y F_Y(A) = 0$. This assumption implies that $A \cap Y = \emptyset$. By definition of clearing schedule, there is a time T at which the dirty set is the empty set. However, this implies that $A \cap Y \neq \emptyset$ or else there would exist a dirty cell. Again we have a contradiction. ∎

Proposition 5.2 *Let* A *be restricted to feasible paths for a given number of searchers* K*. Given a graph* G*,* $\max_{A \subseteq N'} \min_Y F_Y(A) = 0$ *if and only if* $K < s(G)$*, where* $s(G)$ *is the node search number of* G*.*

Proof The value of $s(G)$ implies that a clearing schedule exists for all $K \geq s(G)$. By Proposition 5.1, this implies that a schedule A exists such that $\min_Y F_Y(A) > 0$ for all $K \geq s(G)$.

Similarly, the value of $s(G)$ implies that a clearing schedule does not exist with $K < s(G)$. By Proposition 5.1, this implies that a schedule A does not exist such that $\min_Y F_Y(A) > 0$ for $K < s(G)$. ∎

Proposition 5.1 shows that any non-zero solution to the optimization problem in Equation 5.2 will also be a node clearing solution with K searchers. Proposition 5.2 shows that $s(G)$, the node search number of G, will affect the optimization problem in Equation 5.2; if $K < s(G)$, then $\min_Y F_Y(A) = 0$ for all A. The result from Proposition 5.2 is now used to show that no bounded approximation can exist for the worst-case problem unless $P = NP$. The intuition behind this result is that in order to provide a nontrivial approximation guarantee, an algorithm must achieve nonzero reward whenever the optimal reward is nonzero, which means it must clear the graph if possible.

Theorem 5.1 *Assuming* $P \neq NP$*, there can be no bounded polynomial-time approximation algorithm for the worst-case optimization problem in Equation 5.2.*

[5] Before examining these propositions, the reader may wish to review the definitions of cleared set, clearing schedule, node search number, and recontamination from Section 4.1.

Proof Let n be an arbitrary problem instance defined by the graph input G and number of searchers K. Let $f(n)$ be any strictly positive function of the problem instance (e.g., a function of the number of vertices, number of searchers, graph diameter, etc.). We will prove that if there exists a polynomial-time algorithm that is guaranteed to find $A' \subseteq N'$ such that $\min_Y F_Y(A') \geq f(n) \max_{A \subseteq N'} \min_Y F_Y(A)$, then $P = NP$. Note that $f(n) \leq 1$ for all n, since $\min_Y F_Y(A')$ cannot be greater than optimal.

Given K searchers and a graph $G = (N, E)$, assume that a polynomial-time algorithm $\mathcal{F}$ exists that is guaranteed to generate paths A' s.t. $\min_Y F_Y(A') \geq f(n) \max_{A \subseteq N'} \min_Y F_Y(A)$. Proposition 5.2 showed that $\max_{A \subseteq N'} \min_Y F_Y(A) = 0$ if and only if $K < s(G)$. Thus, we can test $s(G) \leq K$ by testing whether $\min_Y F_Y(A')$ is non-zero. Determining whether $s(G) \leq K$ is a known NP-hard problem (Kehagias et al., 2009a). Thus, $\mathcal{F}$ would solve an NP-hard problem in polynomial time, which would imply $P = NP$. ∎

Theorem 5.1 shows that we cannot expect to generate any approximation guarantee for the worst-case problem in Equation 5.2, and hence the worst-case component of the combined problem.[6] However, as mentioned above, the average-case problem admits a constant factor approximation on the finite-horizon using sequential allocation (see Chapter 3). This analysis motivates the decision to split the searchers into two groups: average-case searchers and worst-case searchers, which preserves some approximation guarantees in the average-case. From Proposition 5.2, we know that at least $s(G)$ searchers must make the worst-case assumption to generate a schedule with any nonzero worst-case reward. These theoretical results motivate the use of guaranteed search algorithms that minimize the attained search number. However, current guaranteed search algorithms in the literature do not minimize clearing time. The next section shows how this limitation can be overcome.

5.2 Combined Search Algorithm

Drawing off the observations in the previous section, an algorithm can be designed that both clears the environment of an adversarial target and performs well with respect to a target motion model. The first step in developing a combined algorithm is to generate a guaranteed search schedule that improves clearing time with a given number of searchers. The Guaranteed Search with Spanning Trees (GSST) algorithm from Chapter 4 is extended to do just this.

The GSST algorithm is an anytime algorithm that leverages the fact that guaranteed search is a linear-time solvable problem on trees. Barrière et al. (2002) showed

[6]It is also interesting to note that the adversarial sensor network placement problem of the same form described by Krause et al. (2007) has the same hardness property. Theorem 5.1 serves as a generalization of their result to the problem of searching for an adversarial target.

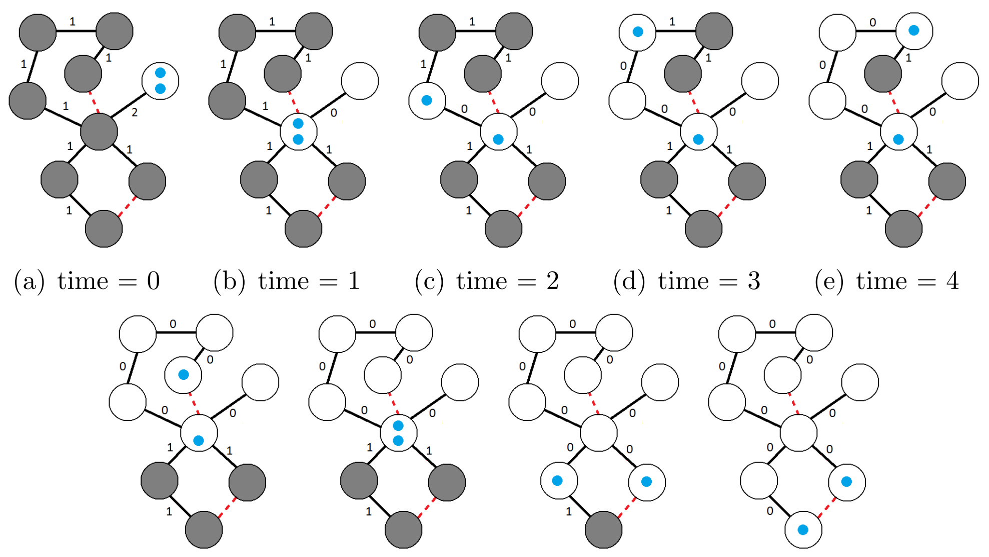
how to generate a recursive labeling of a tree (we will refer to this labeling as B-labeling) in linear-time. Informally, the labeling determines how many searchers must traverse a given edge of the tree in the optimal schedule (a non-recursive B-labeling algorithm is given in Chapter 4). If an edge label is positive, it means that one or more searchers must still traverse that edge to clear the tree. From the labeling, a guaranteed search algorithm can be found using the minimal number of searchers on the tree. GSST generates spanning trees of a given graph and then B-labels them. The labeling is then combined with the use of "guards" on edges that are not in the spanning tree to generate a guaranteed search schedule on arbitrary graphs. In Chapter 4, searchers executing clearing on the underlying tree would call for guards when they could not progress due to non-tree edges. The number of guards was reduced due to the temporal aspects of the search (i.e., guards would not need to remain in place during the entire schedule). Though GSST stores the schedule with the lowest clearing time, it gives no mechanism for utilizing more searchers than the minimal number. Thus, it does not effectively optimize clearing time when given a surplus of searchers.

Algorithm 5.1 shows how GSST can be modified to improve clearing time with additional searchers. The algorithm uses B-labeling to guide different groups of searchers into subgraphs that are cleared simultaneously (see Section 4.3.1 for intuition regarding the use of an underlying spanning tree). The searchers follow the edges with the lowest B-labels, which lead them into areas of the graph that require fewer searchers to clear.[7] Algorithm 5.1 does not explicitly use guards on non-tree edges; the guards are instead determined implicitly from the B-labeling. Thus, it is referred to as Guardless-GSST or G-GSST. Note that the use of B-labeling allows the searchers to perform the schedule asynchronously. For example, a searcher who arrives at a node does not need to wait for other searchers to finish their movement before clearing the next node (assuming that the move does not cause a recontamination).

Figure 5.1 gives an example of clearing with G-GSST. Two searchers clear the simple house graph by following the B-labeling. Searchers are not specifically assigned as guards, and all searchers follow the same movement rules. If a third searcher were added, it would utilize the labeling to clear additional nodes and decrease the clearing time. In contrast, if GSST from Chapter 4 were applied, the third searcher would remain as a stationary guard and would not contribute directly to clearing the graph.

G-GSST can be combined with model-based search algorithms to generate a combined search schedule. G-GSST can be augmented with FHPE+SA to yield a combined search algorithm. In short, the FHPE+SA algorithm has each searcher plan its own path on a finite-horizon and then share that path with other searchers in a sequential fashion (see Chapter 3 for a formal description). In other words, one

[7] In some cases, there may be more than one minimal B-label adjacent to a given node. In this chapter, these ties are broken randomly, which leads to a higher diversity of schedules. Alternatively, the edge could be followed that leads to a higher probability of capture the target. This extension would allow for the clearers to optimize average-case performance to a limited degree.



(a) time = 0 (b) time = 1 (c) time = 2 (d) time = 3 (e) time = 4

(f) time = 5 (g) time = 6 (h) time = 7 (i) time = 8

Figure 5.1: Example of clearing a simple environment represented as a graph using G-GSST. Solid edges denote edges in the spanning tree representation, and dotted edges denote non-tree edges. The searchers (circles) follow the B-labeling (shown next to each edge) to clear the graph. Cells that may contain an adversary at each time are shaded. One searcher remains as an implicit guard in the middle of the graph (any clearing move would cause contamination), while the other searcher moves to clear the top portion of the graph. Once the top portion is cleared, both searchers then progress to clear the bottom of the graph.

Algorithm 5.1 Guardless Guaranteed Search with Spanning Trees (G-GSST)

```
1: Input: Graph $G = (N, E)$, Tree $T = (N, S)$, B-labeling $\lambda(S)$, Searchers $K_g$, Start node $b$
2: $t \leftarrow 0$, $s_k(0) \leftarrow b$ for all $k$
3: **while** $G$ not cleared **do**
4:     **for** all searchers $k$ **do**
5:         **if** moving will recontaminate **then**
6:             $s_k(t+1) \leftarrow s_k(t)$
7:         **else if** $\lambda(e) > 0$ for any $e$ adjacent to $s_k(t)$ **then**
8:             $\mathcal{E} \leftarrow$ edges adjacent to $s_k(t)$
9:             Travel along $e^* = \text{argmin}_{e \in \mathcal{E}} \lambda(e)$ s.t. $\lambda(e) > 0$
10:            $\lambda(e^*) \leftarrow \lambda(e^*) - 1$
11:        **else**
12:            Move towards closest $e'$ with $\lambda(e') > 0$
13:        **end if**
14:    **end for**
15:    **if** $s_k(t+1) = s_k(t)$ for all $k$ **then**
16:        Return failure
17:    **end if**
18:    $t \leftarrow t + 1$
19: **end while**
20: Return clearing schedule $\mathbf{S}$
```

searcher plans its finite-horizon path and shares it with the other searchers; then another searcher plans its finite-horizon path and shares it, and so on. After the initial sequential allocation, searchers replan asynchronously as they reach replanning points in the environment. It is assumed in this chapter that the searchers have all-to-all communication, which allows each searcher to share its plan with any other searcher. This assumption may be relaxed if certain areas of the map can be searched without affecting the schedules of members in other areas (see Chapter 6).

Algorithm 5.2 shows how G-GSST can be augmented with FHPE+SA to yield a combined algorithm. For the remainder of this chapter, searchers executing G-GSST will be referred to as *clearers* and searchers executing FHPE+SA as *optimizers*. An important quality of the combined search is that, depending on the actions of the optimizers, the schedule may be non-monotone (i.e., it may allow for recontamination). However, since the schedule of the clearers is monotone, the search will still progress towards clearing the environment of a worst-case target.

Though not shown in Algorithm 5.2, the schedule of the clearers can be generated in conjunction with the optimizers. For instance, the clearing schedule can be modified based on the actions of the optimizers by pruning portions of the map that happen to be cleared by the optimizers. Implementing this extension requires ensuring monotonicity in the G-GSST schedule, which can be done by dynamically relabeling the spanning tree and taking into account the current cleared set. Since labeling is a linear-time operation, this extension does not significantly affect runtime and provides a tighter coupling between the clearing and average-case optimization.

Finally, many spanning trees are generated (similar to the GSST algorithm) to develop many search schedules (see Algorithm 5.2). These schedules range in quality and also tradeoff worst-case and average-case performance. At any time, the user can stop the spanning tree generation and choose to execute the search schedule that best suits his or her needs. This yields a solution to the worst-case/average-case search problems that, with increasing runtime, continues to generate progressively better solutions.[8]

5.2.1 Decentralized and Online Operation

Once a schedule is found using Algorithm 5.2, the execution can be modified online with decentralized computation. During execution, the optimizers can replan at every iteration based on new information that becomes available (e.g., noisy measurements of the target's positions). In addition, the clearers can continue their schedule, as long as their movement does not cause recontamination, even if the other clearers fall behind in their schedules. Additional replanning is required if the environment map changes online (see Section 5.4.5 for a description of this modification).

[8]Note that the result in Theorem 4.1 does not hold here, and we cannot guarantee that a schedule with minimal clearing time can be generated by G-GSST. The development of such completeness guarantees for G-GSST would require additional randomization, and is left for future work.

Algorithm 5.2 Anytime combined search algorithm

```
1: Input: Graph $G = (N, E)$, Searchers $K$, Maximum computation time $\tau$
2: while computation time left do
3:    Generate spanning tree $T = (N, S)$ and B-labels $\lambda(S)$
4:    for $K_g = K$ down to $K_g = 1$ do
5:       Assign $K_g$ guaranteed searchers and $K - K_g$ efficient searchers
6:       while $G$ not cleared and moves possible do
7:          for all searchers $k$ do
8:             if guaranteed searcher then
9:                Run G-GSST step
10:            else
11:               Run FHPE+SA step
12:            end if
13:         end for
14:      end while
15:      if clearing feasible then
16:         Store schedule
17:      else
18:         Break
19:      end if
20:   end for
21: end while
22: if schedules stored then
23:    Return schedule with maximal $\alpha R_{avg} + (1 - \alpha) R_{worst}$
24: else
25:    Run FHPE+SA to maximize $R_{avg}$ (clearing schedule not found)
26: end if
```

Modifications during execution are possible because of the decentralized nature of the FHPE+SA and G-GSST algorithms. A G-GSST clearing schedule is determined by the spanning tree representation, which is shared by the searchers. The searchers only need to share changes in the cleared set and the current edge labeling to determine their next move. Similarly, the sequential nature of FHPE+SA allows each searcher to plan its own actions and then share that information with its teammates. If the information is not current, or a robot fails, the average-case performance could suffer somewhat, but the search team can still continue operating. Thus, the optimizers are robust to robot failure. In Section 5.4 a decentralized implementation of the combined algorithm is shown on a human-robot search team.

5.3 Complexity and Performance Bounds

This section shows that the average-case component of the combined algorithm is a bounded approximation on the finite-horizon. Let A be the set of time-stamped nodes visited by the paths returned by the FHPE+SA algorithm. Let A^{OPT} be the set of time-stamped nodes visited by the optimal average-case paths (maximizes

Equation 5.1), and let A_k^{OPT} be the set of time-stamped nodes visited by searcher k on the optimal path. Let $F^{AC}(A) = \sum_{Y\in\Psi} \Pr(Y)F_Y(A)$, i.e. the average-case reward for a path A as described in Section 5.1. The average-case reward is bounded as in Theorem 5.2. This bound is simply the FHPE+SA bound for $K - K_g$ searchers.

Theorem 5.2

$$F^{AC}(A) \geq \frac{F^{AC}(A_1^{OPT} \cup \ldots \cup A_{K-K_g}^{OPT}) - \epsilon}{2}, \tag{5.4}$$

where K is the total number of searchers, K_g is the number of searchers used for the clearing schedule, and ϵ is the finite-horizon error ($\epsilon = R\gamma^{d+1}$, where R is the reward received for locating the target, γ is the discount factor, and d is the search depth).

Proof This bound is immediate from the theoretical bounds on sequential allocation (Singh et al., 2007), the monotonic submodularity of F^{AC} (see Chapter 3), and the fact that $K - K_g$ searchers are deployed to perform sequential allocation. The addition of searchers performing G-GSST cannot decrease $F^{AC}(A)$ since clearing an extra node at time t cannot cause additional nodes to become contaminated at time $t' > t$. ∎

The FHPE+SA component of the combined algorithm can be extended to optimize over several different known models. If there are M models being considered, and each has a probability of $\beta_1 \ldots \beta_M$, the weighted average-case objective function in Equation 5.5 can be optimized.

$$F(A) = \beta_1 \sum_{Y\in\Psi} P_1(Y)F_Y(A) + \ldots + \beta_M \sum_{Y\in\Psi} P_M(Y)F_Y(A), \tag{5.5}$$

where $\beta_1 + \ldots + \beta_M = 1$, and $P_m(Y)$ describes the probability of the target taking path Y if it is following model m.

If all models obey the Markov assumption, this linear combination of models can be estimated using matrices as if it were a single model with only a linear increase in computation. Additionally, monotonic submodularity is closed under nonnegative linear combination, so Theorem 5.2 holds in this extended case.

The labeling component of G-GSST requires visiting each node once and is $O(N)$, where N is the number of nodes in the search graph. The G-GSST traversal is $O(NK_g)$, where K_g is the number of guaranteed searchers. Finally, the FHPE+SA component replanning at each step is $O(N(K - K_g)b^d)$, where b is the branching factor of the search graph, and d is the FHPE search depth. Thus, generating a single plan with the combined algorithm is $O(N + NK_g + N(K - K_g)b^d)$. This is linear in all terms except the FHPE search depth, which is fixed beforehand and can be tuned to a very low number depending on the application.

Table 5.1: Statistics for test environments of increasing complexity (attained search numbers found using G-GSST).

	Cave	Office	Museum	Merced	SDR	MOUT
Nodes	43	61	70	130	188	227
Edges	46	65	93	197	258	289
Attained search number	3	3	5	7	8	9

5.4 Combined Search Experiments

5.4.1 Simulated Results

Simulated testing of the combined search algorithm was performed in several complex environments. The Newell-Simon Hall (NSH) and National Gallery (museum) maps were used in previous chapters and were discretized by hand into rooms and hallways. The cave map is taken from the Player/Stage project (Gerkey et al., 2003) and represents a sparsely cluttered environment. The McKenna MOUT map is a polygonal representation of a military testing facility. The cave and MOUT maps were discretized automatically using a constrained Delaunay triangulation (Shewchuk, 2002), which allows for size constraints on the triangles to model limited sensor range. The Merced and SDR maps were used in prior work with the GRAPH-CLEAR problem (Kolling and Carpin, 2008). Kolling and Carpin's method for discretization is not suitable for node search because it does not generate convex regions. The SDR and Merced maps were discretized using a region growing method: each region is grown until its bounding boxes contains a number of obstacle pixels (set to 200). This region growing approach assumes that the evader is large enough that it cannot hide behind small obstacles. In addition, a maximum length for the bounding box was set to 100 pixels to model a limited sensor range. Table 5.1 gives the attained search numbers for these environments (i.e., the fewest number of searchers for which a clearing schedule was found). Figure 5.2 shows discretized maps of the four new test environments (the museum and office are shown in Chapter 4).

5.4.2 G-GSST Comparison

The G-GSST algorithm was tested to determine its effectiveness at generating schedules with low clearing times. G-GSST was compared to a sequential greedy algorithm (RHGC) that iteratively decreases the dirty set on the receding horizon. RHGC was discussed in more detail in Chapter 4.

This section also compares G-GSST to the Parallel Stochastic Hill-Climbing for Small Teams (PARISH) algorithm introduced by Gerkey et al. (2005). PARISH stochastically samples a limited space of possible paths while allowing for recontam-

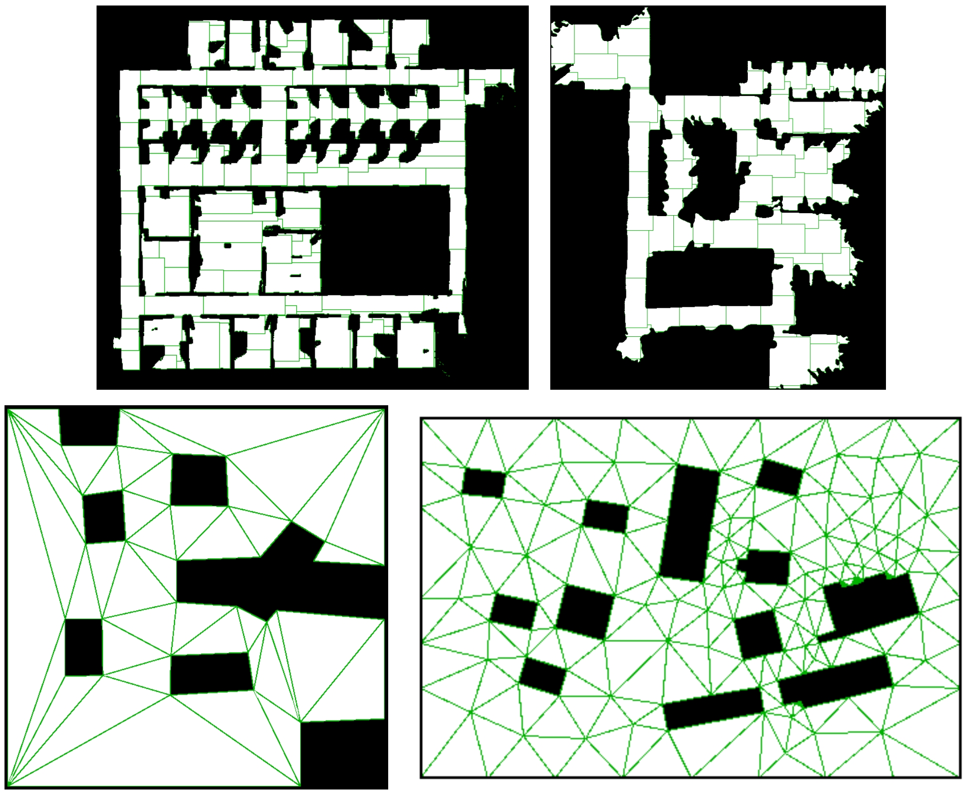


Figure 5.2: Additional maps of environments used for simulated coordinated search. The SDR (top left) and Merced (top right) were discretized using an orthogonal region growing method, and the cave (bottom left) and MOUT (bottom right) were discretized using a constrained Delaunay triangulation.

ination. As a result, the cleared set grows and shrinks until eventually a clearing schedule is found. In addition, PARISH biases the chosen paths towards those that clear more cells. The version of PARISH implemented in this chapter has been modified from the version presented in prior work. The current implementation uses an exponential bias, which places significantly more weight on plans with that clear more cells. The algorithm has also been modified to consider a target located on the nodes rather than on the edges of the graph. Finally, by forcing PARISH to restart whenever a plan is unsuccessful, the algorithm is allowed to run for a given period of time and return the best solution. Together these modifications improve the performance of PARISH over what was presented in prior work (Gerkey et al., 2005).

Figure 5.3 shows a comparison of G-GSST with PARISH and RHGC. A comparison to standard GSST is given in Section 5.4.3. Since PARISH and G-GSST are stochastic, they each were run for one minute in each trial, restarting after each successful or failed schedule. PARISH trials were considered failed after the schedule exceeded 1000 steps without clearing the environment. The best feasible schedule was taken after time ran out. RHGC was run once to completion, which took approximately one second in the larger environments. Even though the runtime of RHGC is lower, it does not provide any mechanism for improving its solution after completion. Stochastic algorithms like PARISH and G-GSST, on the other hand, allow the continued generation of potential schedules with increasing runtime.

In all environments, RHGC required more searchers than G-GSST to find a feasible clearing schedule. As a result, fewer searchers would be available for assignment as optimizers. Furthermore, G-GSST yielded faster clearing times than RHGC in most cases, particularly when only few searchers are available. When many searchers were used, RHGC and G-GSST perform competitively, since tight coordination is no longer required. In addition, G-GSST and RHGC perform competitively on the MOUT map with any number of searchers. This is likely due to the regularity of the triangular decomposition, which somewhat negates the benefit from long-term planning on the spanning tree representation. The triangular decomposition resembles a regular grid, and utilizing the underlying spanning tree does not provide much assistance for generating a clearing schedule for such a topology. These results suggest that utilizing a spanning tree is more beneficial when the graph topology is closer to a tree. In these cases, the spanning tree allows for long-term planning without much increase in computational overhead.

The PARISH algorithm was able to find clearing schedules with few searchers, but the resulting clearing times were very large. The large clearing times are due to PARISH's stochastic nature and its use of recontamination. PARISH's clearing and recontamination hill climbing can take a very large number searcher moves before finding a clearing schedule.

The results above show that G-GSST yields fast clearing times with few searchers, which leads to better worst-case performance as well as more searchers available to improve average-case performance. The effectiveness of utilizing the additional searchers to optimize a target motion model is now examined.

5.4.3 Combined Algorithm Comparison

The combined algorithm was run with an increasing number of random spanning trees on all maps. The target is modeled as randomly moving, and it has an equal chance of moving to any of the adjacent cells every time step (same as in the efficient search trials in Chapter 3). Even in the large maps, the capture times of the best schedules stopped improving significantly after approximately 5000 random spanning trees. This result is somewhat surprising considering that the MOUT map has on the order of 10^{50} spanning trees. Empirically, a small sample of spanning trees is representative of a much larger distribution of schedules and possible clearing times. The high performance after a small number of iterations demonstrates that there are a large number of "good" spanning trees on which to base the search. However, the existence of very rare spanning trees with lower clearing times cannot be ruled out.

The combined algorithm was compared to the pure average-case and pure worst-

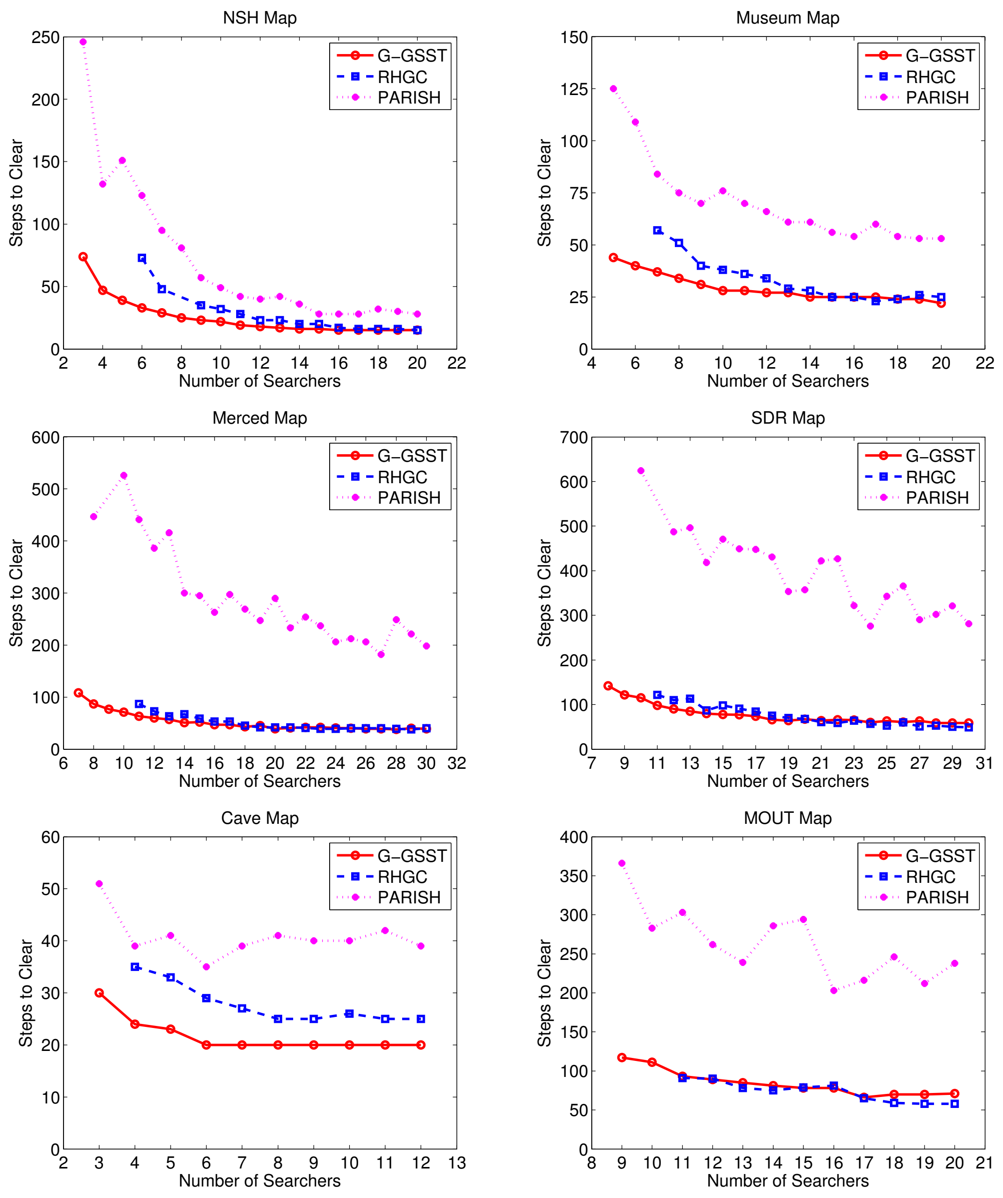


Figure 5.3: Clearing time comparison of the proposed G-GSST algorithm with PARISH (Gerkey et al., 2005) and RHGC (see Chapter 4). G-GSST and PARISH were given one minute to generate solutions, and the best was taken after the time. RHGC was run once to completion. The graph shows the number of steps to clear the graph (i.e., one or more searchers moves in a step) with increasing searchers. Values are only shown if a clearing schedule was found with that number of searchers.

case algorithms. Table 5.2 gives a summary of these results on 2000 random trees. The first row shows the average-case steps (i.e., at each step one or more searchers move between nodes) to capture a randomly moving target using FHPE+SA with all searchers (a lookahead distance of five was used for FHPE on these and all other trials).[9] This strategy does not have worst-case guarantees if the model is incorrect. The second row shows the best clearing time using GSST on 10,000 random spanning trees. Note that GSST does not optimize for clearing time and only utilizes the minimal number of searchers required to clear. The results show that the combined algorithm yields much lower clearing times than those found with GSST. This is due to the use of additional searchers as in Algorithm 5.1. In addition, the combined algorithm reduces the average capture time by over 75% in the office when compared to GSST. Furthermore, a worst-case guarantee on some maps can be gained by sacrificing only one step of average capture time.

The user can determine the weighting of average-case vs. worst-case performance by tuning the value of the α parameter, which explores the Pareto-optimal frontier of solutions. This frontier forms a convex hull of solutions, any of which can be selected at runtime. Figure 5.4 gives a scatter plot of average-case versus worst-case capture steps for all feasible schedules on the six test maps. The figure also shows the number of clearers K_g used for each data point. Note that the total number of searchers is fixed throughout the trials, and the remainder of the searchers perform FHPE+SA. In most cases, the lowest average-case capture times are from the lowest K_g. This is due to more searchers being used for average-case search and fewer for clearing. These results demonstrate the utility of using FHPE+SA searchers in the schedule. Similarly, the lowest clearing times are typically with $K_g = K$ (i.e., all searchers are guaranteed searchers). Note that better solutions yield points to the left/bottom of these plots.

An interesting observation in Figure 5.4 is that the average-case and worst-case times are not perfectly segmented based on searcher allocation (i.e., some strategies with few optimizers perform better than other strategies with more optimizers). This variation between strategies is caused by the selection of spanning trees for the clearing component of the schedules. Some spanning trees complement the average-case clearing schedule more than others. Thus, it is important to consider variations in the clearing schedule even when average-case performance is the primary goal.

Figure 5.5 shows the number of feasible clearing schedules generated per second for various test environments with increasing available searchers. As the number of available searchers increases, the number of schedules generated initially increases as clearing schedules become easier to find. As the number of searchers continues to

[9]The expected capture steps for FHPE+SA were determined over 10,000 trials with a randomly moving target. The expected capture steps for the combined algorithm were computed in closed form, since the environment is cleared. If the environment is cleared at time T and the probability of capture is $\Pr(t)$ for all t, the expected capture steps is $E(T) = \sum_{t=1}^{T} \Pr(t)t$. The probability of capture $\Pr(t)$ was computed using matrix multiplication for a Markovian target model.

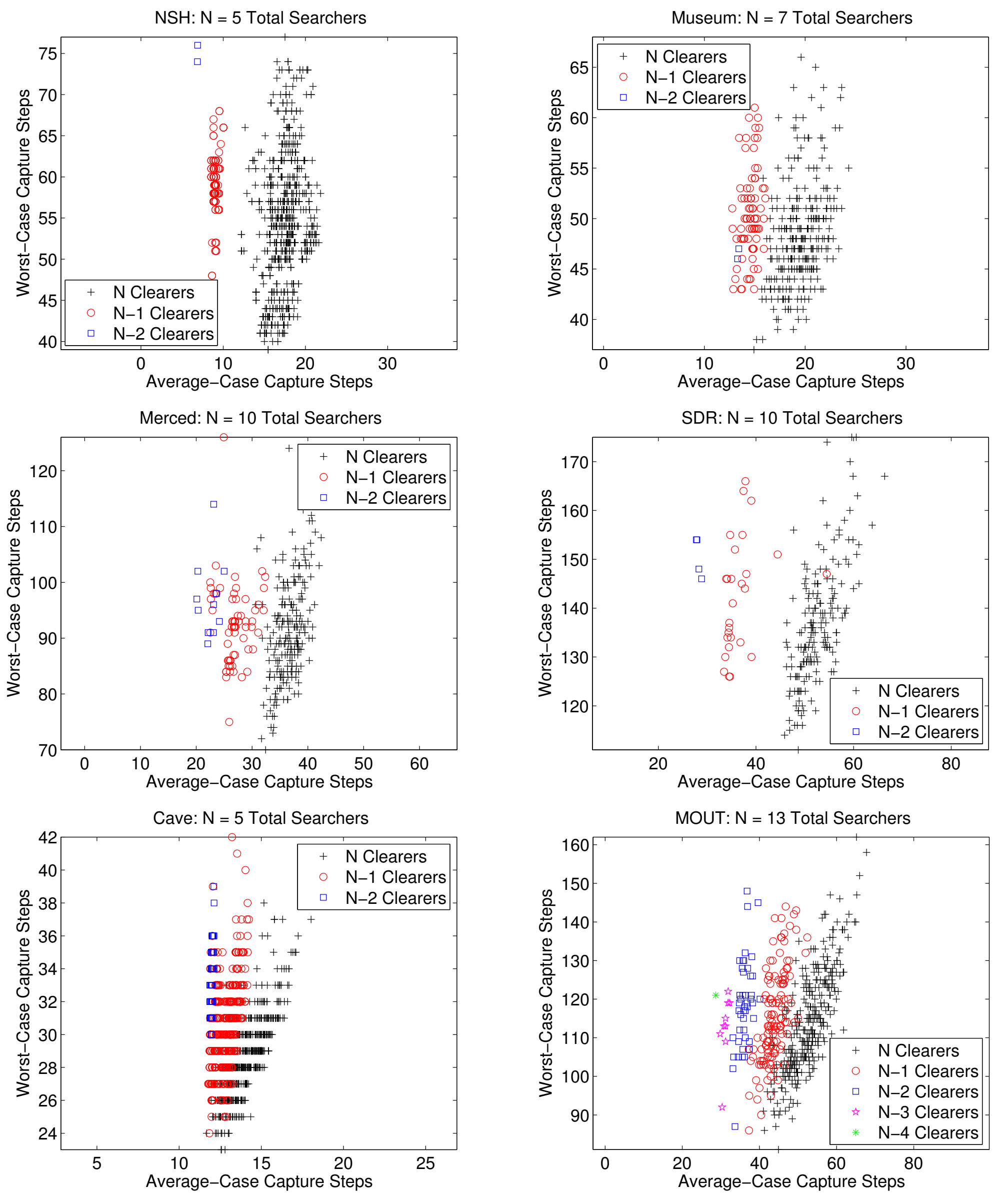
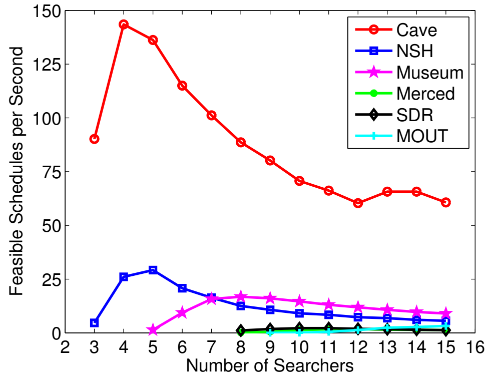


Figure 5.4: Scatter plot of search schedules from the combined algorithm run on 2000 spanning trees. Each data point represents a schedule generated by a different searcher allocation and spanning tree input to Algorithm 5.2. The data points are colored based on the allocation of searchers to the guaranteed and efficient search roles. For instance, $N-1$ clearers means that all robots except one are dedicated to guaranteed search. The total searchers in each environment remains constant throughout.

Table 5.2: Average-case and worst-case capture steps comparison. The average-case (A.C.) is the expected steps to capture a randomly moving target. The worst-case (W.C.) is the number of steps to clear the environment. The proposed combined algorithm is compared to a purely average-case method (FHPE+SA) and a purely worst-case method (GSST).

	Cave ($K = 5$)	NSH ($K = 5$)	Museum ($K = 7$)
FHPE+SA	A.C. 11.1 W.C. ∞	A.C. 5.4 W.C. ∞	A.C. 9.3 W.C. ∞
GSST	A.C. 14.7 W.C. 30	A.C. 24.6 W.C. 74	A.C. 21.8 W.C. 47
Combined ($\alpha = 0.25$)	A.C. 12.6 W.C. 23	A.C. 15.5 W.C. 39	A.C. 14.9 W.C. 37
Combined ($\alpha = 0.75$)	A.C. 11.6 W.C. 24	A.C. 8.6 W.C. 48	A.C 12.9 W.C. 43
Combined ($\alpha = 0.99$)	A.C. 11.6 W.C. 24	A.C. 6.9 W.C. 74	A.C. 12.8 W.C. 51

	Merced ($K = 10$)	SDR ($K = 10$)	MOUT ($K = 13$)
FHPE+SA	A.C. 13.3 W.C. ∞	A.C. 16.5 W.C. ∞	A.C. 22.6 W.C. ∞
GSST	A.C. 45.7 W.C. 112	A.C. 56.0 W.C. 135	A.C. 65.8 W.C. 150
Combined ($\alpha = 0.25$)	A.C. 32.4 W.C. 70	A.C. 48.7 W.C. 111	A.C. 44.9 W.C. 81
Combined ($\alpha = 0.5$)	A.C. 25.9 W.C. 75	A.C. 48.7 W.C. 111	A.C. 33.7 W.C. 87
Combined ($\alpha = 0.75$)	A.C. 25.9 W.C. 75	A.C. 33.5 W.C. 127	A.C 30.3 W.C. 92
Combined ($\alpha = 0.99$)	A.C. 20.1 W.C. 97	A.C. 27.8 W.C. 154	A.C. 28.7 W.C. 121



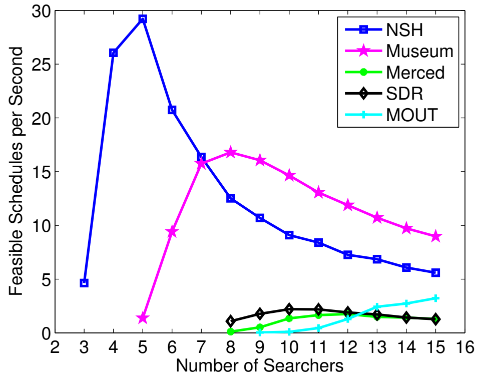


Figure 5.5: Number of feasible clearing schedules per second generated by the combined algorithm with a varying number of searchers. As the number of searchers increases, the number of schedules generated first increases as schedules become easier to find and then decreases as the computational complexity becomes greater. The same graph is shown zoomed out (left) and then zoomed in (right) for additional detail. Schedules with fewer searchers than those displayed were not found after examining 1000 trees on each graph.

increase, the schedules per second decreases due to the added computational complexity. Even in the large environments with few searchers, the combined algorithm is often able to generate more than one feasible clearing schedule per second.

5.4.4 Imperfect Motion Models

In some cases, the target will not act in accordance with the assumed model, but it also will not be fully adversarial. To test the robustness of the combined algorithm in these cases, several complex motion patterns for the target were examined. This section shows that targets moving according to three intentional motion patterns can be found efficiently even when the searchers are utilizing a random motion model to plan their search strategy.

Three target behaviors are introduced: the *daily*, *gallery*, and *escape* behaviors. Targets following the daily behavior act as if they are performing everyday tasks in an office building: they spend significantly more time in offices, they rarely backtrack when moving, and they stop in hallways only for a short time. Targets following the escape behavior act as if they are attempting to find an exit to the building: they never backtrack, rarely return to a room they have already been in, and spend very little time standing still. Finally, targets performing the gallery behavior act as if they are viewing art in a museum: they spend most of their time moving slowly through rooms that they have not yet visited.

The daily, gallery, and escape behaviors are implemented using a set of randomized rules and a memory. The simulated target remembers all locations that it has previously visited, and its actions are determined based on a randomization of the set schedule. The randomized policies are summarized below. They were implemented for a simulated target using a pseudorandom number generator and a complete memory of the target's previous path. The searchers do not have access to the target's true model during planning. None of the above strategies can be fully modeled using a Markovian model because they require remembering history of where the target has been.

Daily Behavior

- if in office (dead-end room): 20% leave office and move to hallway, 80% stay in office
- if not in office: 5% move to last room or hallway visited, 20% move to an office (if one ore more adjacent offices have already been visited, bias towards these offices), 60% keep moving to non-office (exclude hallways previously visited since leaving an office), 15% stay in current hallway

Gallery Behavior

- if in dead-end room: 80% leave this room, 20% stay in this room
- if not in dead-end room: 20% stay in this room, 20% move to last room previously visited, 60% move to a room not visited
- if all adjacent rooms have been visited and not in dead-end: move towards closest unvisited room

Escape Behavior

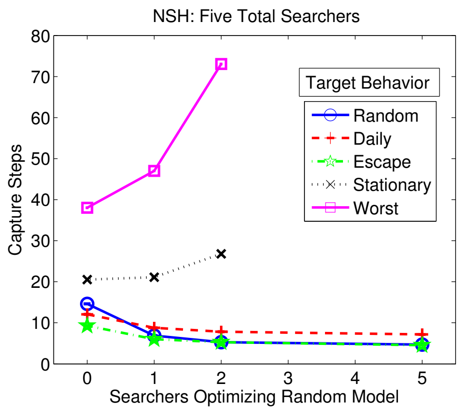
- if in dead-end room: 90% leave this room, 10% stay in this room
- if not in dead-end room: 5% move to last room or hallway previously visited, 30% move to dead-end room (only those never visited), 65% keep moving (only to a room never visited), 0% stay
- if all adjacent rooms have been visited and not in a dead-end: move towards closest unvisited room

The three behaviors described above were implemented on simulated targets in the NSH and museum environments. Searchers perform the combined algorithm assuming a random motion model with weighting parameter $\alpha = 0.5$. Figure 5.6 shows that the daily, gallery, and escape behavior all yield capture times similar to a target truly moving randomly. Thus, searchers that assume a random model will often quickly find targets performing these complex behaviors. For comparison, a stationary target is also considered. The stationary target does not move throughout the environment, which requires the searchers to examine all possible locations. The average time to find a stationary target is closer to the worst-case clearing time than that of the targets moving through the environment. These results are expected because a moving target is, on average, easier to find than a stationary one (see Chapter 3). Note that all schedules, except those assigning all searchers to optimizers, clear the environment, and a target following any model will be found by the time the schedule is completed.

5.4.5 Partially Known Environments

Additional experiments were run to determine the effectiveness of the proposed algorithms in partially known environments. The searchers were given a version of the map with varying percentages of edges incorrect (i.e., an edge that exists in the map was not known to the searcher or an edge that did not exist was believed to exist by the searcher). To facilitate a large variety of maps, some edges were removed from the true version of the world and believed to still exist by the searchers. Other edges remained in the true version of the world, but the searchers believed them to be blocked. The searchers then planned given their current representation of the world. When a node was entered by a searcher, the true state of the edges adjacent to that node became known the searcher, and that information was shared with the rest of the team. The case of limited communication between teammates is not examined here (see Chapter 6), and it is assumed that a shared, though often incorrect, environment map is known to all team members.

For the case of a known, non-adversarial target model, FHPE+SA can be run without modification in partially known environments. The receding horizon allows for the searchers to utilize new information about the map as they receive it online.



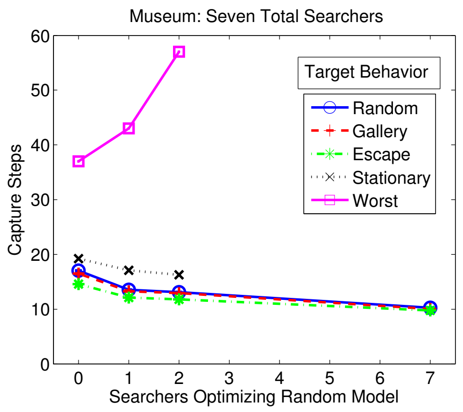


Figure 5.6: Capture times using various models of the target's movement in the NSH and museum environments. A fixed number of searchers was used with some dedicated to clearing and some to optimizing a random motion model. The searchers do not have knowledge of the target's true motion model. Averages were over 10,000 runs, with the target starting in a random location in each run. Standard Errors of the Mean are small and omitted. The stationary and worst-case times do not continue past two optimizers since the environment is no longer cleared.

At each replanning step, the searchers use their current view of the world for both replanning and target modeling. As the true state of the map is revealed, the searchers update their world view for the next replanning iteration. Figure 5.7 shows results from trials with a target moving according to a known random model in partially known versions of the NSH and museums maps. The results show that FHPE+SA still performs well even when very little of the map is known. The results also show that there is a smaller gap between known and partially known environments as the teams size increases.

As shown above, FHPE+SA can be applied directly to partially known environments with a non-adversarial target. However, when a clearing schedule is desired, it is necessary to add a dynamic replanning component to G-GSST. This modification is required because the introduction or deletion of edges can lead to recontamination and invalidate a clearing schedule. Algorithm 5.3 shows an online replanning modification of G-GSST. The extension of G-GSST allows the searchers to replan when they reach a point where they can no longer make progress (due to revealing the true state of the environment). In addition, if progress cannot be made, the searchers will backtrack and attempt to replan. If progress still cannot be made, the searchers will eventually backtrack to the start.

The dynamic replanning version of G-GSST was applied to partially known versions of the NSH and museum maps.[10] The results in Figure 5.8 demonstrate that

[10]Note that removing traversable edges from the NSH map often prevents it from being cleared with three searchers (even if the environment is fully known). Thus, results are reported starting with four searchers.

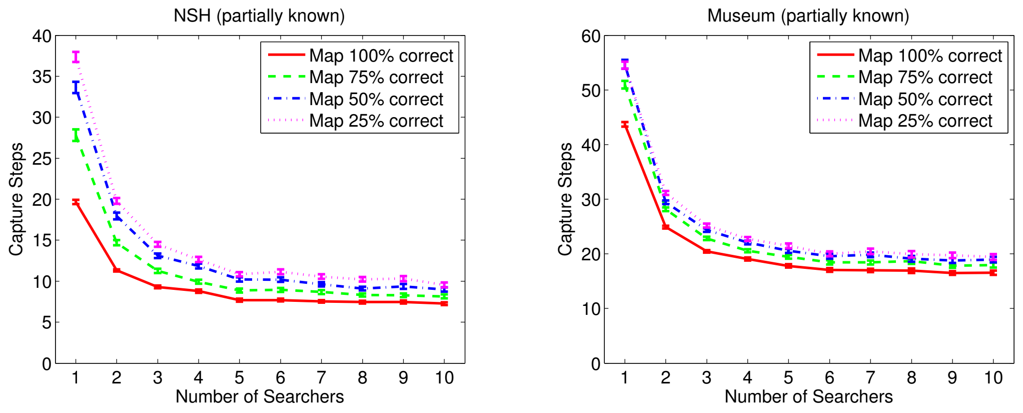
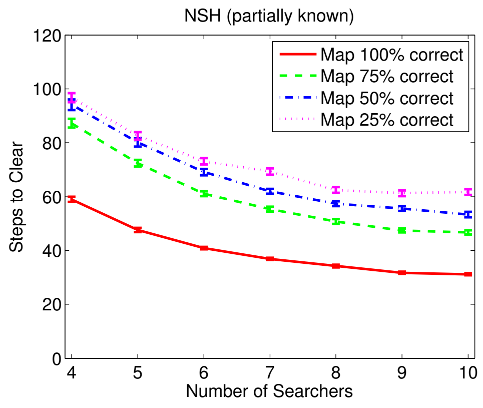


Figure 5.7: Average capture steps in partially known versions of the NSH and museum environments. FHPE+SA was used to optimize locating a randomly moving target. Averages are over 10,000 runs in each of 100 partially known configurations. Error bars are one SEM. Note that all searchers are assigned as optimizers in these results.

Algorithm 5.3 G-GSST with online replanning

```
1: Input: Graph $G = (N, E)$, Searchers $K_g$, Trees to generate $m$, Start node $b$
2: Generate $m$ spanning trees $T$ and G-GSST schedules $\mathbf{S}_T$
3: $T^* \leftarrow \mathrm{argmin}_T\, N_D(t_f|\mathbf{S}_T)$
4: $t \leftarrow 0$, $N_D^{min} \leftarrow N$, $s_k(0) \leftarrow b$ for all $k$
5: while $G$ not cleared do
6:     if $N_D(t_f|\mathbf{S}_{T^*}) \geq N_D^{min}$ then
7:         Backtrack towards $b$
8:         Update $N_D(t+1)$ with any recontamination
9:     else
10:        for all searchers $k$ do
11:            if moving does not cause recontamination then
12:                Execute step of $\mathbf{S}_{T^*}$
13:            end if
14:        end for
15:        Update newly discovered edges in $G$
16:        if $s_k(t) = s_k(t+1)$ for all $k$ then
17:            Generate $m$ more trees $T$ and G-GSST schedules $\mathbf{S}_T$
18:            $T^* \leftarrow \mathrm{argmin}_T\, N_D(t_f|\mathbf{S}_T)$
19:        end if
20:    end if
21:    $t \leftarrow t + 1$
22:    if $N_D(t) < N_D^{min}$ then
23:        $N_D^{min} \leftarrow N_D(t)$
24:    end if
25: end while
```



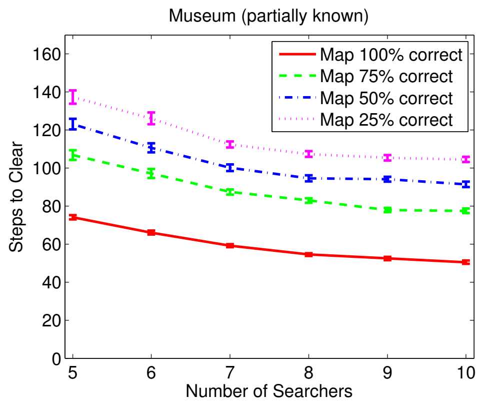


Figure 5.8: Steps to clear in partially known versions of the NSH and museum environments. An extension of G-GSST was used to refine a clearing schedule online. Averages are over 100 partially known configurations. Error bars are one SEM. Note that all searchers are assigned as clearers in these results.

partially known environments lead to at worst a factor of two increase in clearing time, and the gap also tends to decrease as more searchers are added. Thus, both FHPE+SA and G-GSST can be applied to partially known environments with only a small to moderate decrease in performance.

The dynamic replanning version of G-GSST was able to find clearing schedules in all partially known environments using the same number of searchers as the respective fully known environment. However, it may be possible to generate a map that requires more searchers when partially known. For instance, if the searchers reach a point where they believe they cannot make any further progress, they will fail to clear. In these rare cases, Algorithm 5.3 could be augmented with an exploration mode. If the searchers cannot make progress, one or more of them should move to the unexplored nodes. This modification would ensure that any partially known environment does not require more searchers to clear with G-GSST than when it is fully known. However, adding an exploration mode would require careful tuning of the tradeoff between visiting new nodes and clearing what is currently known.

5.4.6 Human-Robot Teams

To examine the feasibility of the combined algorithm for real-world applications, several experiments were run with a human/robot search team. These experiments were conducted on a single floor of an office building as shown in Chapter 4 (Figure 4.8). Two humans and a single Pioneer robot share their position information through a wireless network, and the entire team is guided by a decentralized implementation of Algorithm 5.2 (see Figure 4.9). The robot's position is determined by laser Adaptive Monte-Carlo Localization (Thrun et al., 2005), and it is given waypoints through the Player/Stage software (Gerkey et al., 2003). The humans input their position through the keyboard, and they are given waypoints through a GUI. Each human and robot has knowledge of the true environment map in these trials.

The human-robot search team uses a shared preprocessing step to determine a candidate spanning tree and searcher allocation. Once a schedule is found that balances average-case and worst-case performance, the execution is done online with distributed computation. Each processor plans the actions of a single searcher (human or robot), and the schedule continues executing even if some searchers fall behind. If moving will cause recontamination, and the searcher is a clearer, it will wait until the other searchers catch up before continuing. This avoids strict synchronization of the schedule, which reduces communication between the team. These online modifications are particularly desirable for real-world applications because they avoid many of the robustness issues with executing an offline schedule.

In the first experiment, all three searchers (humans and robot) were assigned as clearers ($\alpha = 0.0$). This configuration took 178 seconds to clear the floor of a potential adversary, and it yielded an expected capture time of 78 seconds w.r.t. the random model. The expected capture time if the model of the target's speed is 50% off (i.e., the target is moving 50% slower than expected) was also calculated. With this modification, the expected capture time increased to 83 seconds. The clearing schedule was executed once, and the expected capture times were calculated in closed form as described above. There was not a physical target in these experiments.

In the second experiment, the mobile robot was switched to an optimizer by using $\alpha = 0.5$. This configuration took 177 seconds to clear and yielded an expected capture time of 73 seconds. The expected capture time with a 50% inaccurate model was 78 seconds. Thus, switching the mobile robot to an optimizer yields a 5 second decrease in expected capture time without sacrificing any worst-case capture time. The 5 second decrease remains even if the model is inaccurate. This further confirms the simulated results in Figure 5.6, showing that the schedules are robust to changes in the target's motion model.

In both experiments, the schedule clears the left portion of the map first and then continues to clear the right portion. In this environment, the assignment of the robot as a clearer does not decrease the clearing time. The reason is that the robot spends a significant portion of the schedule as a redundant guard. Consequently, the right portion of the map is not searched until very late in the clearing schedule. In contrast, when the robot is used as an optimizer, the right hallway is searched early in the schedule, which would locate a non-adversarial target moving in that area.[11]

In addition, the results with a human/robot team demonstrate the feasibility of the communication and computational requirements of the proposed algorithm on a small team. The initial plan generation stage is distributed among the searcher network, and once stopped by the user, the best plan is chosen. During execution, there is a broadcast communication requirement as the searchers share their positions on the

[11]Multimedia extensions (see Appendix A) show playback of the combined algorithm running in both simulated and human-robot trials.

search graph. This small amount of information was easily handled by a standard wireless network in an academic building.

5.5 Chapter Summary

This chapter has presented an algorithm for improving the efficiency of multi-robot clearing by generating multi-agent search paths that both clear an environment of a potential adversary *and* optimize a non-adversarial target motion model. In addition, a new algorithm was introduced for finding clearing schedules with low search times by utilizing the underlying spanning tree to guide clearing. This approach was integrated into a combined algorithm to generate schedules that perform well under both average-case and worst-case assumptions on the target's behavior.

It was shown through simulated experiments that the combined algorithm performs well when compared to search algorithms for both guaranteed and efficient search. The proposed method was validated using imperfect models and partially known environments. In addition, the feasibility of the algorithm was demonstrated on a heterogeneous human/robot search team in an office environment. Combining search guarantees against an adversarial target with efficient performance using a model has the potential to improve autonomous search across a wide range of applications. This chapter has contributed the first algorithm and results towards this goal.

The results in this chapter further confirm implicit coordination's ability to solve tightly coordinated multi-robot search tasks. The combined algorithm preserves the idea of allowing each robot to plan its own path, while taking into account information from the rest of the team. The main difference from previous chapters is that some robots are assigned to operate on the spanning tree representation and others to optimize a target motion model. The combined algorithm highlights the modular nature of implicit coordination. Robots performing different tasks can work together as long as each robot knows what actions the rest of the team is planning to perform, and the effect of those actions can be incorporated into the robot's own plan.

Chapter 6

Incorporating Connectivity Constraints

THIS chapter considers the problem of multi-robot coordination subject to constraints on the configuration. In many scenarios, communication will be limited by the environment, which requires the team to maintain line-of-sight, range, and/or other constraints to share information, coordinate, and check for failures. Building on the problems discussed in previous chapters, the case is examined in which a mobile network of robots must search, cover, or survey an environment while remaining connected. The addition of connectivity constraints increases the coupling between the robots, since the feasibility of a given robot's path is now dependent on the paths of the other robots. As shown in the guaranteed search domain, tight coupling can lead to poor performance of implicitly coordinated solutions. However, it is demonstrated that relaxing the connectivity constraints using the idea of *periodic connectivity*, where the network must regain connectivity at a fixed interval, allows for the application of implicit coordination to the constrained search domain as well.

Unlike in previous chapters, the constrained planning problem will be formulated as a general multi-robot informative path planning problem (MIPP) with non-adversarial search as a special case. This highlights the broader applicability of some of the techniques described in this thesis. The MIPP problem is extended to incorporate periodic connectivity, which is referred to as MIPP-PC. An extension of FHPE+SA (see Chapter 3) is proposed for planning with periodic connectivity that scales linearly in the number of robots and allows for arbitrary connectivity constraints. Theoretical hardness results and performance guarantees are shown, and the approach is validated in the coordinated search domain in simulation and in real-world experiments. The proposed algorithm significantly outperforms a gradient method that requires continual connectivity and performs competitively with a market-based approach, but at a fraction of the computational cost.[1]

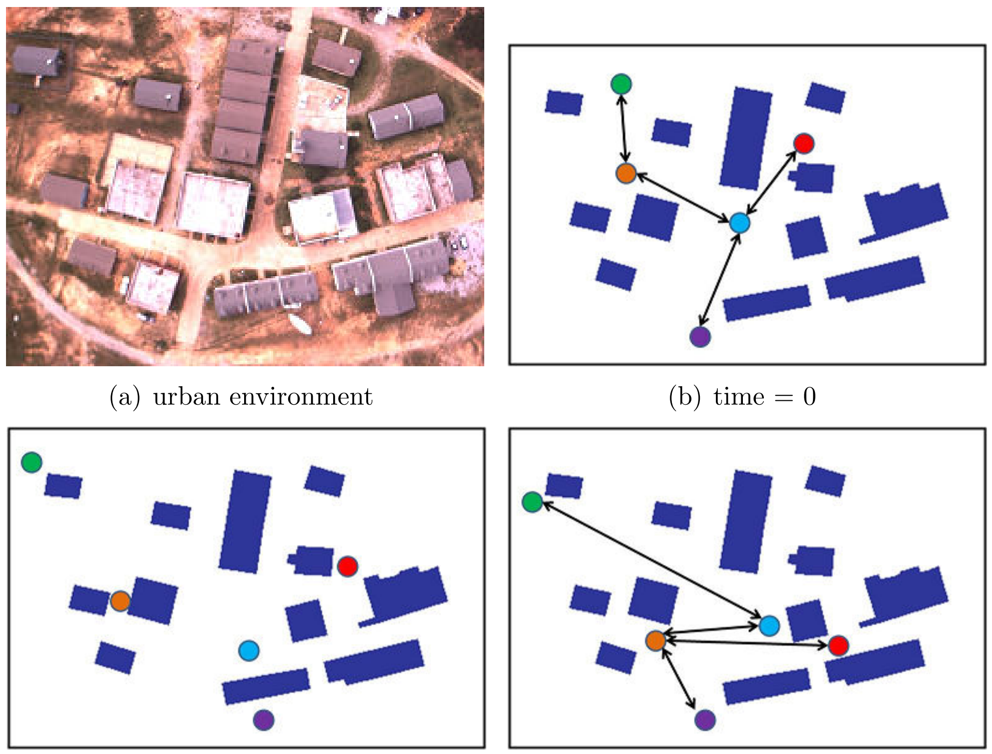
[1]The algorithm proposed in this chapter was originally introduced by Hollinger and Singh (2010).

To complement the active coordination algorithm, this chapter also describes a passive technique for fusing data for robots performing search tasks that have become disconnected for a significant period of time. In many multi-robot search applications, all-to-all communication cannot be guaranteed for the duration of the mission, which motivates the design of techniques for merging disparate data when the searchers become reconnected. Prior work has introduced a decentralized data fusion framework that allows for such merging under strict assumptions on the underlying distribution. However, these techniques are not directly applicable to search tasks, due to the necessity of modeling a multi-modal estimate of the target's position (e.g., the target may be in several locations but not anywhere between them). A data fusion method is presented in this chapter that integrates into the decentralized data fusion framework and is applicable to both efficient and guaranteed search. The proposed method uses a *minimum fusion rule* to merge differing estimates of the target's position, which is shown to provide a conservative estimate when applied to efficient search. The minimum fusion rule is validated in the efficient search domain through simulated testing. It is shown that decentralized data fusion significantly improves performance relative to cases where a fusion rule is not used. In many cases, the minimum fusion rule is nearly as effective as full communication.

6.1 Constrained Search Problem Setup

A mobile network of robots must plan paths through an environment to gain information, or even simply to get from one place to another. This general problem includes a broad set of applications that require decentralized planning. This includes searching for a mobile target, estimating climate change (Singh et al., 2009a), mapping an environment while remaining localized (Sim and Roy, 2005), and exploring multiple locations (Kalra, 2006). In some applications, the network may be able to communicate in any configuration, which allows for unconstrained planning and coordination of the team. However, in many cases, there are constraints on the network that prevent communication. For instance, obstacles or distance may prevent two robots from communicating.

Prior work in multi-robot coordination often enforces *continual* connectivity constraints. These constraints may be soft (breakable at a penalty) or hard (not breakable at all), but in either case, the goal is to maintain connectivity to all nodes in the network at all times. This chapter introduces the idea of *periodic* connectivity. In many scenarios, it may be desirable to break connectivity if the network plans to become connected in the future. Periodic connectivity allows the network to (1) communicate information gathered, (2) coordinate the next phase of the plan, and/or (3) check that all robots are still operational. The scope of this chapter is limited to information gathering tasks that fit into the MIPP framework, but periodic constraints could potentially apply to any multi-robot domain.



(a) urban environment (b) time = 0

(c) time = 3 (d) time = 5

Figure 6.1: Example of searching a simulated McKenna MOUT site (a), while maintaining periodic connectivity. The buildings prevent line-of-sight communication links (shown as arrows) between robots in the network (shown as circles). The robots start in a connected configuration (b); the network becomes completely disconnected and explores a small part of the environment (c); the network regains connectivity in a new configuration to replan and share information (d).

6.1.1 MIPP-PC Formulation

A team of K robots is given the task of planning paths through an environment that maximize some measure of informativeness. It is assumed that the environment is known and has been discretized into a graph $G = (N, E)$ in which the nodes represent convex regions, and the edges represent connections between these regions (see previous chapters for examples).

The searchers plan in the space specified by the graph $G' = (N', E')$, which is a time-unfolded version of the graph G (see definition in Chapter 3). Each node in N' is a time-stamped copy of a node in N, i.e. each node $n' \in N'$ corresponds to (n, t) for a node $n \in N$ and time t. Each robot k plans a path $A_k = \{r_k(0), \ldots, r_k(T)\}$ for discrete time steps $t = 0, \ldots, T$, where $r_k(t) \in N'$. Let $C(A_k)$ be the cost of the path for robot k. It is assumed that all moves have cost one, so $C(A_k) = T$ for a horizon T; however, an extension to more complex cost functions is fairly straightforward. The union of paths for all robots represents which nodes are visited at which time and is given by $A = A_1 \cup \ldots \cup A_K$. Planning in this space allows modeling non-stationary reward functions as well as revisiting nodes several times in the path. For this chapter, it is assumed that robots can occupy the same cell, and that low-level collision avoidance is available. Alternatively, collision avoidance could be incorporated by disallowing robots from occupying the same cell at the same time.

The robots' goal is to plan paths that maximize an objective function $F(A)$. It is assumed that F is a known, deterministic function of the possible paths. Possible objective functions include discounted probability of capture of a target, mutual information in a Gaussian Process (Singh et al., 2009a), and information gain in a tracking scenario. The MIPP optimization problem is given below:

Problem 6.1 *Given K searchers, time unfolded graph $G' = (N', E')$, and ending time T:*

$$A^* = \max_{A_k \subseteq N'} F(\cup_{k=1}^{K} A_k); \ \textit{s.t. } C(A_k) \leq T, \ \textit{for all } k \in \{1, \ldots, K\}$$

In addition, a graph $G_C = (N, E_C)$ is given that describes the connectivity between the nodes in G (i.e., node u is connected to node v iff there exists an edge $uv \in E_C$).[2] If the graph G_C is not given, but a more general form of connectivity constraints are available (e.g., range constraints or line-of-sight), G_C can be computed during preprocessing. No restrictions are made on the edges in graph G_C, which allows for any type of connectivity constraints. However, it is assumed that connectivity is either "on" or "off," and does not vary within a node. A configuration $A(t)$ is considered connected at time t if there exists a path in G_C from $r_i(t)$ to $r_j(t)$ for all i, j in which all nodes in the path are occupied.

For the MIPP-PC problem, the robots begin in a connected configuration $A(0)$ and must regain a connected configuration at times $T_I, 2T_I, \ldots, \tau T_I$, where τ is the smallest integer such that $\tau T_I > T$. The variable T_I is the *disconnected interval* during which the network may choose to lose connectivity. In many cases, T_I will be dictated by the application, and in other cases it can be selected by the user. When $T_I = 1$, the problem is equivalent to planning with continual constraints. The resulting MIPP-PC problem is given below:

Problem 6.2 *Given K searchers, time unfolded graph $G' = (N', E')$, connectivity graph $G_C = (N, E_C)$, disconnected interval T_I, and ending time T:*

$$A^* = \max_{A_k \subseteq N'} F(\cup_{k=1}^{K} A_k); \ \textit{s.t. } C(A_k) \leq T \ \textit{for all } k \in \{1, \ldots, K\}, \ \textit{and}$$
$$A(\tau T_I) \ \textit{is connected on } G_C \ \textit{for all } \tau \in \{0, \ldots, T/T_I\}$$

[2] In some cases connectivity may change over time, which would require a time-indexed connectivity graph $G_C(t)$. It is straightforward to incorporate this extension by modifying the constraints as the team plans ahead.

Given a fully connected graph G_C, the MIPP-PC problem is equivalent to the MIPP problem, and since MIPP is NP-hard (Singh et al., 2009a), MIPP-PC is NP-hard. Thus, we cannot expect to solve MIPP-PC optimally with an efficient algorithm (unless $P = NP$). This motivates the design of approximation algorithms and heuristics for various instantiations of MIPP-PC.

6.2 Connectivity Constrained Planning Algorithm

Algorithms are now designed for solving the MIPP-PC problem approximately. First, it is shown that the MIPP-PC problem becomes significantly easier if the connected configurations are pre-specified. In this case, algorithms from MIPP can be utilized to solve MIPP-PC. For unspecified connected configurations, a receding horizon implicit coordination algorithm is designed that estimates good connected configurations as well as informative paths between them.

6.2.1 Pre-Specified Connected Configurations

Here it is assumed that the configurations $A(T_I), A(2T_I), \ldots, A(\tau T_I)$ are given either by an oracle or as part of the problem. For instance, robots may need to plan paths between pre-specified checkpoints where the checkpoints provide connected configurations. In this case, the MIPP-PC problem reduces to several instances of the unconstrained informative path planning problem:

Theorem 6.1 *Given the connected configurations $A(0), A(T_I), \ldots, A(\tau T_I)$, multi-robot informative path planning with periodic constraints (MIPP-PC) reduces to τ instances of unconstrained multi-robot informative path planning (MIPP).*

Proof The robots start in an initial configuration $A(0)$, and they must attain a configuration $A(T_I)$ at time T_I. The robots must maximize a function $F(\cup_{t=0}^{T_I} P(t))$ given a budget constraint on their paths $B = T_I$ and path constraints on the graph G'. This is equivalent to the MIPP problem with known starting and goal configurations and a path-constrained budget. The robots must solve τ instances of the MIPP problem to fill in the full path $A(0), \ldots, A(\tau T_I)$. ■

Based on the reduction in Theorem 6.1, algorithms from the MIPP domain can be used to yield good approximations in the MIPP-PC domain with pre-specified connected configurations. For cases where a long lookahead is required and significant computation is available, the branch-and-bound algorithm from Singh et al. (2009a) can be applied. For cases where an online approach is desired, the FHPE+SA algorithm from Chapter 3 can be utilized.

6.2.2 Unspecified Connected Configurations

In many informative path planning scenarios, the connected configurations will not be specified beforehand. A scalable algorithm is now presented for estimating good connected configurations as well as informative paths. The algorithm utilizes receding-horizon planning and implicit coordination to remain tractable in complex environments and with large teams. Each robot maximizes its reward on a horizon that is equal to the disconnected interval, and chooses a path that is connected to the planned configuration of its teammates at the end of the interval. The connectivity restrictions require stricter synchronization than in the unconstrained domain (see Chapter 3), and as a result, asynchronous operation becomes difficult. The proposed algorithm below utilizes synchronous planning to generate high-performing strategies.

Algorithm 6.1 gives a description of the proposed method for determining informative paths with unknown connected configurations. This algorithm is called at the start and then each time the robots reach a connected configuration as specified in the previous plan. Thus, it runs online and can account for new information added at each connected configuration. The computation can be divided among the robots, but it requires synchronized updates of shared paths, which restricts planning to one robot at a time. Since the robots are guaranteed to be connected every T_I, synchronized replanning can occur at this time.

The proposed algorithm can be seen as a coordinate ascent in the space of robot paths (i.e., a method that optimizes one variable at a time in a round-robin while the others are fixed). The first robot plans assuming that all robots will remain stationary, and it ensures that its goal remains connected to its stationary teammates. The first robot shares its plan with the other robots, and they can then plan their paths such that they regain connectivity with the new configuration. Planning order is selected randomly, and it is assumed that there is no connection between order and network neighborhoods. More sophisticated ordering strategies could also be employed, including the use of neighbors in the network. The coordinate ascent continues until the paths no longer change. Since the amount of reward collectable is finite, the number of possible paths is finite, and the reward collected by the planned paths is monotonically increasing, the algorithm will always terminate as long as the robots always have up-to-date information from the rest of the team. In practice, the outer loop can be run only a single iteration and still yield good performance.

Algorithm 6.1 is similar to the FHPE+SA algorithm proposed in Chapter 3 for the unconstrained efficient search domain. However, there are several key differences that allow its extension to the MIPP-PC domain. First, Algorithm 6.1 limits the planned paths to those that regain connectivity at T_I, which reduces the planning space and ensures periodic connectivity. In addition, FHPE+SA replans at every iteration, which is not feasible in the MIPP-PC domain because of connectivity breaks. Furthermore, FHPE+SA runs only a single iteration of planning instead of continuing to convergence. In the MIPP-PC domain, increasing the number of planning iterations allows for a piggyback effect where the planned paths of the robots even-

Algorithm 6.1 Implicit coordination for MIPP-PC

```
 1: Input: robots $K$, graph $G$, connectivity graph $G_C$, objective $F$, interval $T_I$, initial config $A(0)$
 2: % Initialize all paths to remain stationary
 3: $A_k(1,\ldots,T_I) \leftarrow A_k(0)$ for all $k$
 4: % $V \subseteq N'$ is the set of visited time-stamped nodes
 5: $V \leftarrow \cup_{k=1}^{K} A_k$
 6: while not converged do
 7:     for all robots $k \in \{1,\ldots,K\}$ do
 8:         % Reset path for robot $k$
 9:         $V \leftarrow V \ / \ A_k$
10:         Enumerate some feasible paths $B_k(1,\ldots,T_I)$
11:         Discard paths disconnected at $T_I$
12:         $B_k^*(1,\ldots,T_I) \leftarrow \text{argmax}_{B_k} F(V \cup B_k)$
13:         % Update path for robot $k$
14:         $A_k \leftarrow B_k^*$, $V \leftarrow V \cup B_k^*$
15:     end for
16: end while
17: Return $A(0), A(1), \ldots, A(T_I)$
```

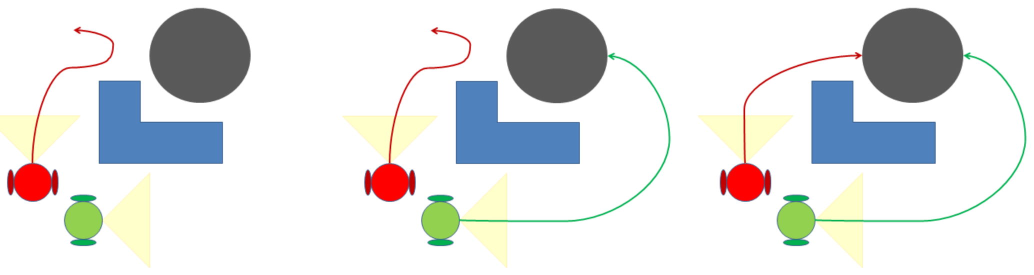


Figure 6.2: Example of implicit coordination with periodic connectivity. The robots (green and red) must move around the obstacle (blue L-shape) to observe the area of high information gain (gray circle). They start in line-of-sight contact, and they must regain line-of-sight past the obstacle. The red robot first plans a path that remains connected to the green robot's initial position (left). Then the green robot plans a path that regains connectivity with the red robot past the obstacle (middle). Finally, the red robot replans to regain connectivity with the green robot's new path (right).

tually converge to a point far away from the start (see Figure 6.2 for an example). Note that Algorithm 6.1 assumes that planning to the next connected configuration is completed before execution starts. This restriction is not necessary, and robots could continue planning during execution as long as it does not jeopardize reconnecting with the team.

6.2.3 Running Time

Each iteration of the outer loop in Algorithm 6.1 requires K optimizations of a single path MIPP-PC instance. If all paths are enumerated to depth T_I, the computation is $O(Kb^{T_I})$, where b is the branching factor of the search graph. If T_I is too great to enumerate all paths, a stochastic sampling of paths can be employed. In practice, only a few iterations are required for convergence, and in many cases good performance is achievable with only a single iteration (see Section 6.4). As noted above, the algorithm can be interrupted to yield a solution at any time after the first iteration.

The application of implicit coordination to MIPP-PC does not allow for the same type of asynchronous operation as in the efficient search domain. Since the searchers are not always able to communicate, they must plan synchronously when they are connected in order to guarantee that all robots have current information about their teammates' plans.

6.3 Analysis of Connectivity Constrained Planning

It is now shown that many multi-robot optimization problems subject to connectivity constraints are inapproximable (i.e., no polynomial-time algorithm can yield a multiplicative performance guarantee unless $P = NP$). It is first proven, through a reduction from the connected coverage problem (Anisi, 2009), that determining if a connected path exists between an initial configuration and a goal configuration is NP-complete. This implies the inapproximability of any multi-robot optimization problem that requires finding a connected path between locations as a subproblem, which includes multi-robot informative path planning.

As stated above, K robots and a graphical representation of an environment $G = (N, E)$ are given. The nodes N represent locations, and the edges E represent traversable links between locations. A graph $G_C = (N, E_C)$ is also given that describes the "connectivity" of the nodes. For instance, a node n_1 may be connected to a node n_2 if there is line-of-sight between the locations represented by these nodes. For the robots to form a connected network, there must exist at least one path in G_C from every occupied node in N to every other occupied node in N.

6.3.1 Connected Coverage

To begin the analysis, the NP-complete connected coverage problem as introduced by Anisi (2009) is now stated. It will then be shown that connected coverage is a subproblem of a large class of multi-robot optimization problems with connectivity constraints.

If we assume that K robots must visit the sets $V_1, \ldots, V_K$ respectively, and we let $V_i \subset N$ denote disjoint subsets of N, for $i \in \{1, \ldots, K\}$, the connected coverage problem is as follows:

Problem 6.3 *Find a configuration $Q = \{q_1, \ldots, q_K\}$, such that $q_i \in V_i$ for all $i \in \{1, \ldots, K\}$, and Q is connected on G_C.*

Anisi gives a polynomial reduction from an arbitrary SAT instance to the connected coverage problem requiring $U + L + 1$ robots and sets, where U is the number of clauses, and L is the number of literals in the SAT instance.

6.3.2 Finding a Connected Path

It is now shown that a more general multi-robot connected path planning problem is NP-complete using a reduction from connected coverage.

Problem 6.4 *Given K robots with starting locations $\mathcal{S} = \{s_1, \ldots, s_K\}, s_i \in N$, and a maximum time T, determine if a set of K-robot paths A exists from the starting locations to a set of goal locations $\mathcal{G} = \{g_1, \ldots, g_K\}, g_1 \in N$, such that A is a feasible K-robot traversal of G, A is connected on G_C at all steps, and the path can be traversed in less than T steps.*

Theorem 6.2 *Problem 6.4 is NP-complete.*

Proof It is easy to show that the problem is in NP. Given a guess of the path, the starting and ending locations, as well as the traversal conditions, can easily be checked. Connectivity checking can be achieved in time linear in the number of robots (and path size) using a breadth first search at each point in the path.

To show that the problem is NP-hard, we give a reduction of the connected coverage problem. Assume w.l.o.g. that a K-robot connected coverage problem is given with sets $V_1, \ldots, V_K$. Generate distinct arbitrary starting locations $\mathcal{S} = \{s_i, \ldots, s_K\}$ and distinct arbitrary goal locations $\mathcal{G} = \{g_i, \ldots, g_K\}$. For each robot i, generate a traversability link between s_i to g_i through the corresponding set V_i. More precisely, add an edge in G between each node $q_i \in V_i$ to both s_i and g_i. In addition, add a connectivity link in G_C between all s_i and s_j, such that the start set is a connected configuration. Also add a connectivity link between all g_i and g_j, such that the goal set is a connected configuration.

If a feasible connected path exists between $\mathcal{S}$ and $\mathcal{G}$, the robots will have solved the connected coverage problem in the intermediary step. If a feasible connected path does not exist, then the intermediary step does not have a connected coverage. Finally, this reduction requires a total of K robots and $2K + |X|$ total nodes, where $X = \cup_{i=1}^{K} V_i$, showing this to be a polynomial reduction. ∎

6.3.3 Multi-robot Path Optimization with Connectivity

A similar reduction to connected coverage can also be used to show that MIPP is inapproximable when continual connectivity constraints are considered:

Theorem 6.3 *Unless $P = NP$, there cannot exist any polynomial-time approximation algorithm for the MIPP-PC problem (Problem 6.2) with $T_I = 1$. More precisely,* *let n be an arbitrary problem instance defined by the inputs G_C, G', and number of searchers K. Let $a(n)$ be any strictly positive function of the problem instance (e.g., a function of the number of vertices, number of searchers, graph diameter, etc.). If there exists a polynomial-time algorithm that is guaranteed to find paths B such that $F(B) \geq a(n) \max_A F(A)$, then $P = NP$.*

Proof We construct an instance of this problem in which any feasible path with nonzero reward will answer an arbitrary connected coverage query (Problem 6.3). Given a K-robot connected coverage problem, generate a start set $\mathcal{S}$ and a goal set $\mathcal{G}$ with traversability and connectivity links as described in the proof of Theorem 6.2. Now, let the function $F(A)$ be positive if and only if at least one robot reaches its goal in $\mathcal{G}$. Due to the construction of the connectivity links, any robot reaching its goal implies that *all* robots reach their goals. It will be important to note that this construction allows for $F(A)$ to be both submodular and nondecreasing as described in Section 6.3.5. We now have an instance of Problem 6.2.

Assume that a polynomial-time approximation algorithm $\mathcal{F}$ exists for the instance constructed above, and that this algorithm has returned a set of paths B. We can easily test whether $F(B)$ is positive. Further, $F(B)$ is positive if and only if the robots reach the goal set (as constructed above). Thus, any reward greater than zero will imply that a connected set of paths exists between the start and goal sets. Since $\mathcal{F}$ is guaranteed to provide some reward if any reward is possible, we can solve the embedded connected coverage problem in polynomial time by testing if $F(B)$ is positive. Connected coverage was shown to be NP-complete by Anisi (2009). Hence, the existence of algorithm $\mathcal{F}$ implies $P = NP$. ∎

Theorem 6.3 shows that adding connectivity constraints prevents us from deriving any polynomial-time approximation guarantee for the MIPP problem with connectivity constraints (unless $P = NP$). This is somewhat surprising because the MIPP problem is relatively easy to approximate without connectivity constraints.

6.3.4 Periodic Connectivity

The analysis above requires that the network remain connected at all times. This constraint can be relaxed such that the network needs to regain connectivity only periodically during execution. It is now shown that this does not change the inapproximability of the multi-robot optimization problem with connectivity constraints.

Problem 6.5 *Equivalent to Problem 6.4 except that A is required to be connected on G_C every T_I steps, where $0 < T_I < T$.*

Theorem 6.4 *Problem 6.5 is NP-complete.*

Proof Use the same reduction as in Theorem 6.2, but make a chain of $T_I/2$ copies of the embedded connected coverage problem. The same proof path applies. ∎

Theorem 6.5 *Unless $P = NP$, there cannot exist any polynomial-time approximation algorithm for MIPP-PC (Problem 6.2) with $T_I > 1$.*

Proof Modify the construction in Theorem 6.3 using $T_I/2$ copies of the embedded connected coverage problem. The same proof path applies. ∎

Note the construction in the proof of Theorem 6.3 requires that an instance of the MIPP problem is created with a connected coverage problem embedded inside. If this is not possible in a specific problem domain, due to special cases of connectivity constraints, then the inapproximability result may not apply.

6.3.5 Performance Guarantees

For this section, the underlying objective function F is assumed to be nondecreasing and submodular on the ground set of time-stamped locations in the graph G' (see Chapter 3). It is also assumed that $F(\emptyset) = 0$. This assumption is necessary to show performance guarantees, however, it does not change the hardness result described above (see the proof of Theorem 6.3).

Submodularity is the formalization of an intuitive diminishing returns quality (i.e., the more locations visited, the less incremental benefit from visiting new locations). Submodularity holds for many interesting reward functions including information gain and discounted reward in most cases (see Section 6.1 for examples).

Let $A_k^{SA}(0, \ldots, \tau T_I)$ be the path found for robot k by the sequential application of a single-robot MIPP algorithm with an approximation guarantee of κ to some end time τT_I (i.e., τ instances of periodic connectivity with disconnected interval T_I). For instance, exhaustive enumeration of paths yields the optimal single-robot path, and achieves $\kappa = 1$. Let $A_k^{OPT}(0, \ldots, \tau T_I)$ be the optimal path for robot k. For the case of known connected configurations, we have the following performance guarantee:

Theorem 6.6 *Sequential allocation on the K-robot submodular, nondecreasing MIPP-PC problem with pre-specified connected configurations achieves a lower bound of:*

$$F(\cup_{k=1}^{K} A_k^{SA}(0, \ldots, \tau T_I)) \geq \frac{F(\cup_{k=1}^{K} A_k^{OPT}(0, \ldots, \tau T_I))}{1 + \kappa} \tag{6.1}$$

Proof Singh et al. (2009a) showed that any κ approximation of a single-robot instance of the MIPP problem on a submodular, nondecreasing function can be extended to a $\kappa + 1$ guarantee for the multi-robot MIPP problem using sequential allocation. Applying Theorem 6.1, we have τ instances of MIPP in which the reward and bound is additive. ∎

In contrast, with unknown connected configurations, a bound can only be achieved containing a constant offset term. Let ρ be the amount of reward outside the horizon

$T_I/2$. For example, if the cost function F is discounted by γ, and R is the total possible reward, $\rho = R\gamma^{T_I/2+1}$. The Theorem below shows bounds in the case of unknown connected configurations:

Theorem 6.7 *Algorithm 6.1 on the K-robot submodular, nondecreasing MIPP-PC problem with unspecified connected configurations achieves a lower bound of:*

$$F(\cup_{k=1}^{K} A_k^{SA}(0,\ldots,\tau T_I)) \geq \frac{F(\cup_{k=1}^{K} A_k^{OPT}(0,\ldots,\tau T_I)) - \rho}{1+\kappa} \tag{6.2}$$

Proof By construction, the robots always start in a connected configuration. Regardless of connectivity constraints, the robots can always return to this configuration. This gives $T_I/2$ steps to perform unconstrained sequential allocation. By applying Theorem 6.6, the bound is immediate. ∎

The bound in Theorem 6.7 depends heavily on the constant offset imposed by ρ, which is determined by T_I. For long disconnected intervals (as $T_I \to \infty$), the guarantee approaches the bound in Theorem 6.6. For short disconnected intervals, the guarantee degrades fairly quickly. It may be tempting to look for algorithms that eliminate the constant offset, but combining Theorem 6.7 with the result in Theorem 6.5 shows that this is unlikely. Removing the constant offset would create a constant factor approximation to the MIPP-PC problem, which is only possible if $P = NP$ from Theorem 6.5.

The quality of the approximate solution relates closely to the accuracy with which the connected configurations are determined. If they are chosen optimally, then Algorithm 6.1 achieves the bound in Theorem 6.6. However, if they are chosen poorly, the bound degrades to that in Theorem 6.7. Since Algorithm 6.1 performs a sequential coordinate ascent, it can be expected to perform well in cases where the constraints and cost function can be smoothly interpolated (i.e., they have a contour to follow towards the optimal).

6.4 Connectivity Constrained Experiments

The proposed algorithm was tested in the non-adversarial search domain in which a team of robots must find a moving target with a known (or approximately known) motion model. To map the formulation from Chapter 3 into the MIPP-PC domain, the cost function F is set to Equation 3.2, with $\gamma = 0.95$, and the disconnected interval $T_I = 5$. Thus, the robots must maximize the discounted probability of capture as well as achieve a connected network every five time steps. The cases of both distance and line-of-sight constraints are examined. In the following experiments, the target estimation distributions are merged optimally; however, for large disconnected intervals the conservative merging technique described later in this chapter could be employed.

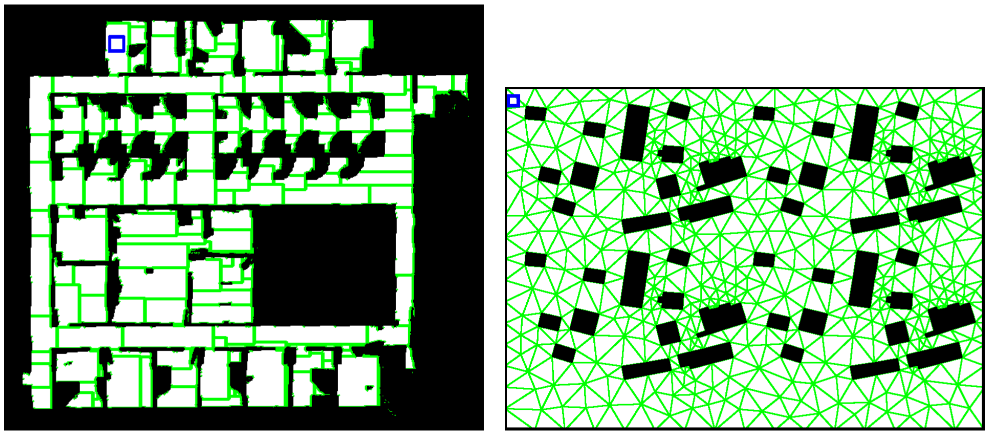


Figure 6.3: Environments used for search trials with periodic connectivity. The SDR office (left) contains 188 cells and was discretized using a region growing technique. The McKenna MOUT site (right) contains 937 cells, and was discretized using a constrained Delaunay triangulation. Cell boundaries are shown as light green lines, and starting cells are denoted with a blue square.

6.4.1 Simulated Experiments

The proposed MIPP-PC algorithm was tested in the environments shown in Figure 6.3. The first environment is a map of the SDR building used in Chapter 5. On this map, the robots are required to maintain periodic range constraints. The range constraints were set as 1/4 the diagonal of the map. The second environment is a repeated version of the McKenna MOUT site tesselated into 937 cells using a constrained Delaunay triangulation. Here, the robots must maintain periodic line-of-sight constraints. The buildings serve as obstacles to obstruct line-of-sight. The map discretization includes only free space; the obstacles are not valid nodes on the map. Obstacle avoidance is handled trivially by moving between the centroids of the convex cells. It is assumed that multiple robots can exist in the same cell at the same time by taking advantage of low-level collision avoidance.

For comparison, several algorithms are introduced similar to those proposed in prior work for continual constraints:

1. Coupled gradient - enumerate a single step forward in the joint planning space and discard all paths that are disconnected. Note that gradient-based methods inherently cannot account for periodic connectivity because they do not plan more than one step ahead. They must maintain continual connectivity otherwise they risk remaining disconnected.

2. Gradient+Fiedler - compute the coupled gradient augmented with a connectivity term represented by the second eigenvalue of the Laplacian matrix (the Fiedler value). This term has been found to provide a good measure of the connectivity of the network (Michael et al., 2008).

3. Market coordination - The robots initialize their paths with a single iteration of Algorithm 6.1. They then compute a two-robot path for each team member in the joint planning space and auction these plans to other robots. This active coordination approach is in the spirit of the Hoplites architecture (Kalra, 2006).

4. Unconstrained baseline - run the FHPE+SA algorithm (see Chapter 3) without periodic constraints. The network is not guaranteed ever to become connected after the first step in the plan. This serves as a baseline because the algorithms that require periodic connectivity will yield higher capture times.

5. Implicit coordination - the proposed approach in Algorithm 6.1. The algorithm is run until convergence.

Figure 6.4 shows simulated results using up to ten robots for the methods described above. The metric used is average-capture steps, where the robots are assumed to move one cell in a step. This intuitive metric relates to discounted capture probability. The disconnected interval remains fixed throughout these experiments. If the interval were reduced to close to one, the problem would resemble continual connectivity, which would likely result in smaller gains from the proposed algorithm over one-step approaches. If the disconnected interval were increased, exhaustive path enumeration may not be possible. However, the proposed algorithm could still be utilized using a stochastic sampling technique, which would maintain the benefits of periodic connectivity.

The coupled gradient performs poorly in SDR as the number of searchers increases, which is due to the algorithm falling into a local maximum that it cannot escape. Incorporating the Fiedler value removes this local maximum but sacrifices performance, because the Fiedler heuristic does not directly relate to the objective function. On the MOUT map, both gradient-based approaches perform poorly because they cannot reason about periodic connectivity. On both maps, the market-based approach performs somewhat better than implicit coordination (on average 10% better in SDR and 5% better in MOUT), but requires over fifteen times more computation with ten robots. In addition, the gap between the MIPP baseline and implicit coordination with periodic connectivity is smaller in SDR (17% on average) than in MOUT (35% on average). This is likely due to periodic constraints preventing the team from spreading out in the larger environment. The gap shrinks as the team size grows, demonstrating the ability to spread into a chain topology.

Figure 6.4 shows the computation time to generate a 50-step plan for the various algorithms. The gradient approaches act in the joint space; they start very fast but scale exponentially in the number of searchers. The market-based approach requires $O(K^2 b^{2d})$ computation to generate a combinatoric number of two-robot joint plans. It also has a higher overhead due to the computation required to resolve the auctions. The proposed implicit coordination approach requires very few iterations to converge and is approximately linear in the number of searchers, thus requiring very little computation even as the team size increases.

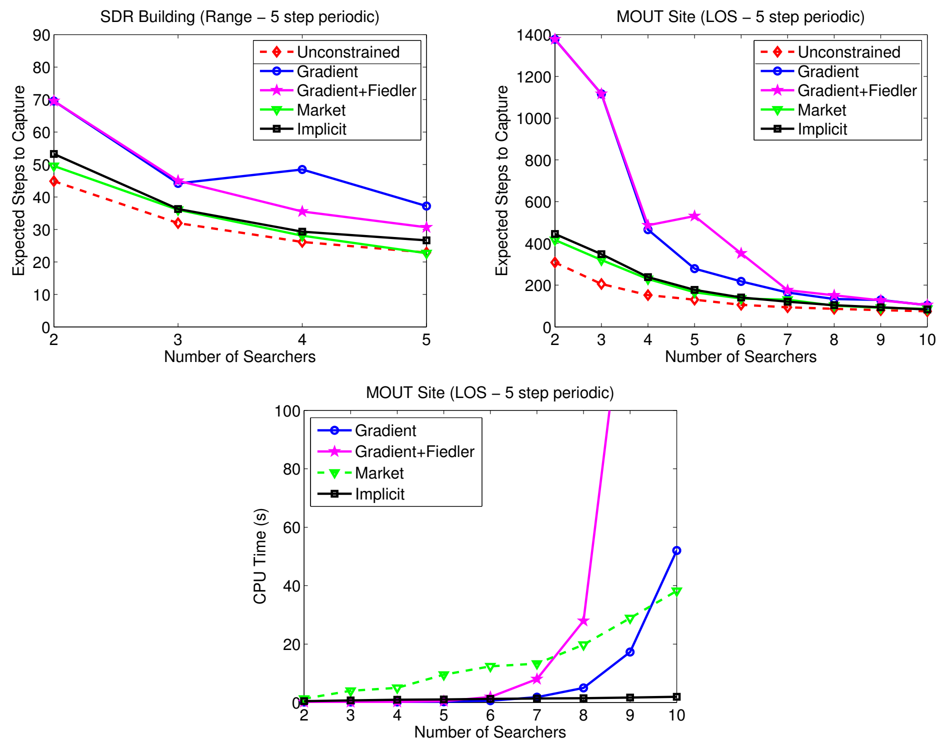


Figure 6.4: Simulated comparison of various methods for multi-robot search with periodic connectivity. The proposed implicit coordination method outperforms gradient-based methods in both the SDR (top left) and MOUT (top right) environments. Market-based coordination provides slightly better capture times than implicit coordination, but requires significantly more computation to generate a 50-step plan (bottom). Expected time to capture is calculated by assuming the target moves with a random walk.

6.4.2 Mobile Robot Experiments

The proposed MIPP-PC algorithm was implemented on a team of two mobile robots: the Serf and Sideswipe autonomous platforms. Each robot plans its own path and maintains its own localization estimate. The inter-robot communication was handled using the Player software (Gerkey et al., 2003), and the robots communicate on a wireless network. Ultra-wideband ranging radios were utilized for localization (Djugash et al., 2006) in an outdoor urban environment. The environment was automatically discretized into 52 cells using a constrained Delaunay triangulation (Shewchuk, 2002). Several obstacles were present, and they were assumed to impede line-of-sight connectivity (see Figure 6.5).

The robots' task was to search for a stationary intruder using a laser scanner while maintaining periodic connectivity. This task is equivalent to the non-adversarial search problem described above, but assuming a stationary target. The goal is to locate a stationary target quickly, which differs from the task of covering or clearing an area quickly. For instance, it may be beneficial to search a high probability location

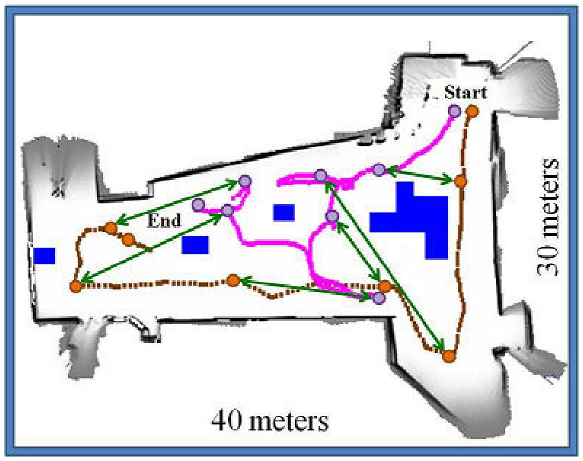


Figure 6.5: Left: The paths of two robots (magenta and brown) searching for a stationary target in an outdoor urban environment. The blue obstacles prevent line-of-sight connectivity. The robots regain connectivity approximately every 30 seconds at the locations connected with green arrows. Right: Two robots round a corner to regain line-of-sight connectivity while searching.

early even if doing so delays the time at which all areas will have been searched. To incorporate these considerations, an efficient search formulation was used. This task was chosen for the robot experiments because the large amount of variation in capturing a mobile target would require many trials to assess performance. In contrast, the average capture time for finding a stationary target can be evaluated using a single trial.

The robots calculated a plan using Algorithm 6.1 and located the target in the center of the map in approximately three minutes. The robots regained connectivity every five steps on the graph (approximately every 30 seconds). Figure 6.5 shows the paths from this experiment. The one-step coupled gradient was run for comparison, and it was unable to find the target due to becoming trapped in a local maximum in which neither robot could make progress without breaking connectivity. Utilizing the Fiedler value did not help here since the network only contained two robots. Other heuristics may improve performance of the one-step approach, but they would require careful tuning of the reward function.[3]

6.5 Data Fusion for Limited Communication

To complement the active coordination techniques described above, a passive data fusion rule is now described for efficient and guaranteed search. The proposed data fusion rule allows searchers that have been disconnected for a long period of time

[3]Multimedia extensions (see Appendix A) show playback of both the simulated and mobile robot trials with periodic connectivity.

to fuse information when they later become reconnected. The proposed fusion rule is consistent with other data fusion techniques found in prior work (Makarenko and Durrant-Whyte, 2004).

The general decentralized data fusion framework estimates some feature of interest (e.g., a target's location) described by a state vector $\mathbf{x}_t$, where t denotes the current time. The feature is modeled using a probabilistic state transition $\Pr(\mathbf{x}_t|\mathbf{x}_{t-1})$, which is assumed to be Markovian. Observations $\mathbf{z}_t$ are received that provide information about the state $\mathbf{x}_t$. A model of the sensor likelihood function is also known that provides $L(\mathbf{z}_t|\mathbf{x}_t)$ given the state at the time of the observation. The Bayesian filtering problem is to find a posterior estimate $\Pr(\mathbf{x}_t|\mathbf{Z}^t, \mathbf{x}_0)$ given the history of observations $\mathbf{Z}^t$ and an initial state estimate $\mathbf{x}_0$. Using the recursive Bayes' rule, calculation of the posterior estimate takes the following form:

$$\Pr(\mathbf{x}_t|\mathbf{Z}^t, \mathbf{x}_0) = \eta L(\mathbf{z}_t|\mathbf{x}_t) \Pr(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{Z}^{t-1}, \mathbf{x}_0), \tag{6.3}$$

where η is a normalizing constant.

Equation 6.3 can be separated into a predictive component in which $\Pr(\mathbf{x}_t|\mathbf{x}_{t-1})$ is applied, and an information fusion component in which $L(\mathbf{z}_t|\mathbf{x}_t)$ is applied. The calculation of Equation 6.3 becomes a decentralized data fusion problem when different nodes i and j (e.g., searchers or robots) receive different measurement histories $\mathbf{Z}_i^t$ and $\mathbf{Z}_j^t$, and wish to reconcile them into a common estimate $\Pr(\mathbf{x}_t|\mathbf{Z}_{i\cup j}^t)$. In this case, there is some redundant information between the estimates $\Pr(\mathbf{x}_t|\mathbf{Z}_{i\cap j}^t)$. If the redundant information is known, the fused estimate can be calculated in closed form (Makarenko and Durrant-Whyte, 2004). However, in many cases the redundant information is not known because it has already been folded into the estimate. Thus, recovering the true distribution can require storing a large number of measurements and reapplying the filtering steps. For increasingly large teams, the combinatorics of such perfect fusion becomes infeasible.

In the case where the redundant information is not known, it is desirable to develop a conservative estimate of fused distribution. A conservative estimate always overcorrects for the redundant information (possibly undercounting it). For the case of Gaussian distributions, it is possible to achieve a conservative estimate using a weighted combination of the disparate estimates and covariances (Julier and Uhlmann, 2001). However, particularly in search applications, the target's distribution cannot be modeled using these assumptions. A method is presented below that provides a conservative fusion method for search applications using very little computation.

6.5.1 Efficient Search Data Fusion Problem

The problem of data fusion is now formulated for efficient search. As described in Chapter 3, the target's state at time t is represented as a belief vector $b(t) = [b_0(t), b_1(t), \ldots, b_N(t)]$, where $b_0(t)$ is the probability that the target has already been found at time t, and $b_1(t)$ through $b_N(t)$ are the probabilities that the target is in cells 1 through N respectively. The belief vector $b(t)$ describes probability that the target is in each possible state of $\mathbf{x}_t$ in the decentralized data fusion framework.

A "capture" event is represented by the application of a matrix C to the belief vector, where C moves some probability from the cells 1 through N to the capture state. A capture event is an observation of the target's state and corresponds to a measurement $\mathbf{z}_t$. The target's movement is modeled using a matrix D that disperses the probability based on a motion model but does not move any probability to the capture state. Subsequent capture and dispersion events are modeled by multiplying the target's belief vector by additional matrices. The dispersion matrices apply the motion model $\Pr(\mathbf{x}_t|\mathbf{x}_{t-1})$.

If searchers cannot communicate and later become reconnected (e.g., they loose line-of-sight connectivity for a while), they will want to share the information they have gained. However, it may be computationally prohibitive to store their entire measurement history and then reapply the measurement vectors. In addition, if multiple searchers come together at the same time, the necessary merging could require the application of a large number of dispersion and capture matrices. An alternative is to apply a conservative merging rule that does not require the reapplication of the matrices.

6.5.2 Minimum Fusion Rule

Before introducing a fusion rule, we will require some additional notation. When searchers become disconnected, it is assumed that each of them maintains their own belief vector based on the application of their own capture matrices and the capture matrices of other searchers with which they are in contact. Each searcher k's local belief vector will be referred to as $b^k(t)$. The belief vector that would result if all capture matrices were applied will be referred to as $b^*(t)$. If the searchers have full communication, $b^k(t) = b^*(t)$ for all k. The following rule is introduced to merge the estimate of searcher i with any number of searcher estimates into a single merged estimate $b^m(t)$:

$$b_n^m(t) = \min_j b_n^j(t), \tag{6.4}$$

where $\min_j$ is chosen over all searchers within communication range of searcher i. Equation 6.4 is applied over all cells n. After applying the rule in Equation 6.4, the belief vector must be renormalized. If a standard renormalization is applied (i.e., divide all elements by the total probability), the value of the capture state could actually be reduced. This approach is undesirable since the target will never escape capture. An alternative that avoids this drawback is to adjust the capture state directly to reflect the estimates in each cell after the application of the minimum rule:

$$b_0^m(t) = 1 - \sum_{n=1}^{N} b_n^m(t) \tag{6.5}$$

If multi-hop communication is allowed, the resulting distribution $b^m(t)$ is shared by all searchers connected to searcher i. However, if only single-hop communication is possible, different searchers may maintain different estimates even after a merge, due to having different neighbors. In this case, each searcher's merged distribution contains the minimum probability that a target is in each cell given its immediate neighborhood. However, after several subsequent merges, the multi-hop information will be propagated through the network and all connected searchers will eventually share the same distribution.

The intuition behind this fusion rule is that each observation will reduce the amount of probability in the environment. Thus, the searchers will want to regain that information through the fusion rule if the observation is missed. The next section will show that the merged distribution has several desirable properties relative to the true distribution.

6.5.3 Analysis of Fusion Rule

It will now be shown that the minimum fusion rule never overestimates the probability that a target is captured, and never underestimates the probability that a target is in a cell. Overestimating capture would lead to search schedules that avoid areas that would be searched in the optimal schedule, which is undesirable. The following assumptions are made for this analysis:

Assumption 6.1 *The initial belief $b^i(0)$ equals $b^j(0)$ for all searchers i and j. All searchers start with the same belief over the target's state.*

Assumption 6.2 *The dispersion matrix D is known to all searchers. All searchers have the same model of the target's behavior.*

Theorem 6.8 shows that the minimum fusion rule will never underestimate the probability that a target is in a given cell.

Theorem 6.8 *The value $\min_k b_n^k(t)$ is greater than $b_n^*(t)$ for any cell n, any time t, and any number of searchers included in $\min_k$.*

Proof The argument is that the capture matrices C_j and the dispersion matrix D are monotone on the appropriate subvectors of the target's belief (i.e., the capture state and non-capture states respectively). The proof will be by induction on $b_n(t)$. By assumption, $b_n^k(0) = b_n^*(0)$ for all searchers k and cells n. We now show that $b_n^k(1) \geq b_n^*(1)$.

Let α_{ij} be the probability that a target moves from cell i to cell j, which is encoded in the dispersion matrix D. Note that $\sum_i \alpha_{ij} = 1$, for all j. Let β_i^j be the probability that a target is captured in cell i given that capture matrix C_j is applied.

The application of a capture matrix C_j can be rewritten for each cell n as:

$$b_n(t+1) = (1 - \beta_n^j)b_n(t) = b_n(t) - \beta_n^j b_n(t). \tag{6.6}$$

The dispersion rule can be rewritten for each cell n as:

$$b_n(t+1) = \alpha_{n1}b_1(t) + \ldots + \alpha_{nN}b_N(t). \tag{6.7}$$

By assumption, each searcher applies the same dispersion matrix D to the belief vector. After the application of the dispersion matrices $b_n^k(0^+) = b_n^*(0^+)$. If communication is perfect, all capture matrices will be applied to $b^k(0^+)$ that are applied to $b^*(0^+)$, which leads to $b^k(1) = b^*(1)$.

Due to imperfect communication, one or more capture matrices may not be applied. W.l.o.g. assume that capture matrix C_j is the first capture matrix applied to $b^*(0^+)$ that is not applied to $b^k(0^+)$. Now, $b_n^k(1^-) = b_n^*(1^-) + \beta_n^j b_n^*(1^-) \geq b_n^*(1^-)$. For each subsequent capture matrix applied or not applied, this inequality continues to hold. Thus, in all cases $b_n^k(1) \geq b_n^*(1)$.

We now continue with the induction on $b_n^k(t)$. If $b_n^k(t) = b_n^*(t)$, then the argument above holds for $b_n^k(t+1)$. If $b_o^k(t) > b_o^*(t)$ for any o, then the application of the dispersion matrix to $b_n^k(t)$ is a linear combination of lesser values than those used for $b_n^*(t)$. In this case, $b_n^k(t^+) > b_n^*(t^+)$. For each capture matrix applied or not applied, the inequality continues to hold as above. Thus, $b_n^k(t+1) \geq b_n^*(t+1)$ for all n. The same argument can be applied to all k. ∎

An immediate corollary is that no searcher will ever overestimate the belief that the target is found.

Corollary 6.1 *The value $b_0^k(t)$ is less than $b_0^*(t)$ at any time t.*

Proof Immediate from the fact that $b_0^*(t) = 1 - \sum_{n=1}^N b_n^*(t)$ and Theorem 6.8. ∎

The analysis above shows that the searchers will never become "more sure" of the target's capture due to using the minimum merging strategy. In addition, the searchers will never believe that a target is not in a cell when it actually has a high likelihood of being there. Thus, the resulting strategy will be conservative. Some areas may be searched more often that they would be without the merging, but no area will be neglected due to the merging.

The minimum fusion rule is not the only conservative fusion rule possible in this domain. Any rule that avoids overestimating the probability that the target is in a given cell will also be conservative. For instance, a rule that averages over the beliefs of all adjacent searchers will yield a conservative estimate. However, the minimum rule provides a better estimate of the true distribution than the average rule. To see why, recall from the analysis above that missed application of capture matrices will always increase the amount of probability in a given cell. Thus, the average rule throws out more information about capture than the minimum fusion rule. In addition, the average rule is more difficult to implement since the resulting distribution changes if it is applied multiple times (e.g., averaging over two robots that are connected and then averaging with a third yields a different result than averaging over all three at once).

6.5.4 Simulated Data Fusion Experiments

Experiments were run to test the effectiveness of the proposed fusion rule in three environments with either range or line-of-sight constraints on communication. The searchers utilize FHPE+SA to plan paths that maximize the discounted probability of finding the target. Searchers are allowed to coordinate with the other searchers within the communication constraints. Multi-hop communication is disallowed in these experiments, and searchers can coordinate only with their immediate neighbors. In addition, searchers who cannot communicate do not share state and do not apply each other's capture matrices. If the merging rule is used, searchers that regain communication merge their estimates as in Equations 6.4 and 6.5. Note that, unlike in the active coordination results above, the searchers do not plan to come in contact with each other; they simply maximize the probability of detection by coordinating and communicating with searchers within the communication constraints.

Recall that the efficient search objective function is:

$$J(U(1),\ldots,U(T)) = \sum_{t=0}^{T} \gamma^t \Pr(\exists i : s_i(t) = e(t)), \tag{6.8}$$

where $U(1),\ldots,U(T)$ are the searchers' planned paths, γ is a discount factor, $s_i(t)$ is the location of searcher i at time t, and $e(t)$ is the location of the target at time t. $\Pr(\exists i : s_i(t) = e(t))$ is equivalent to the capture state $b_0(t)$ at time t. Applying Corollary 6.1, we see that the minimum fusion rule is a conservative estimate relative to this objective function (i.e., the expected reward using the minimum fusion rule will always be an underestimate of the reward using the true distribution).

Figure 6.6 shows the effectiveness of using the minimum merging rule to locate a randomly moving target in the six environments from Chapter 5.[4] In all environments, the merging rule decreases the expected time to capture the target. In many cases, the conservative merging rule yields expected capture times nearly as low as if full communication were available. The one exception is the line-of-sight constraints

[4]In these results, the target has a 1% chance of moving to each adjacent cell, and otherwise remains stationary. This model represents a more slowly moving target than in previous experiments, which better highlights the effect of incorporating history through the merging rule.

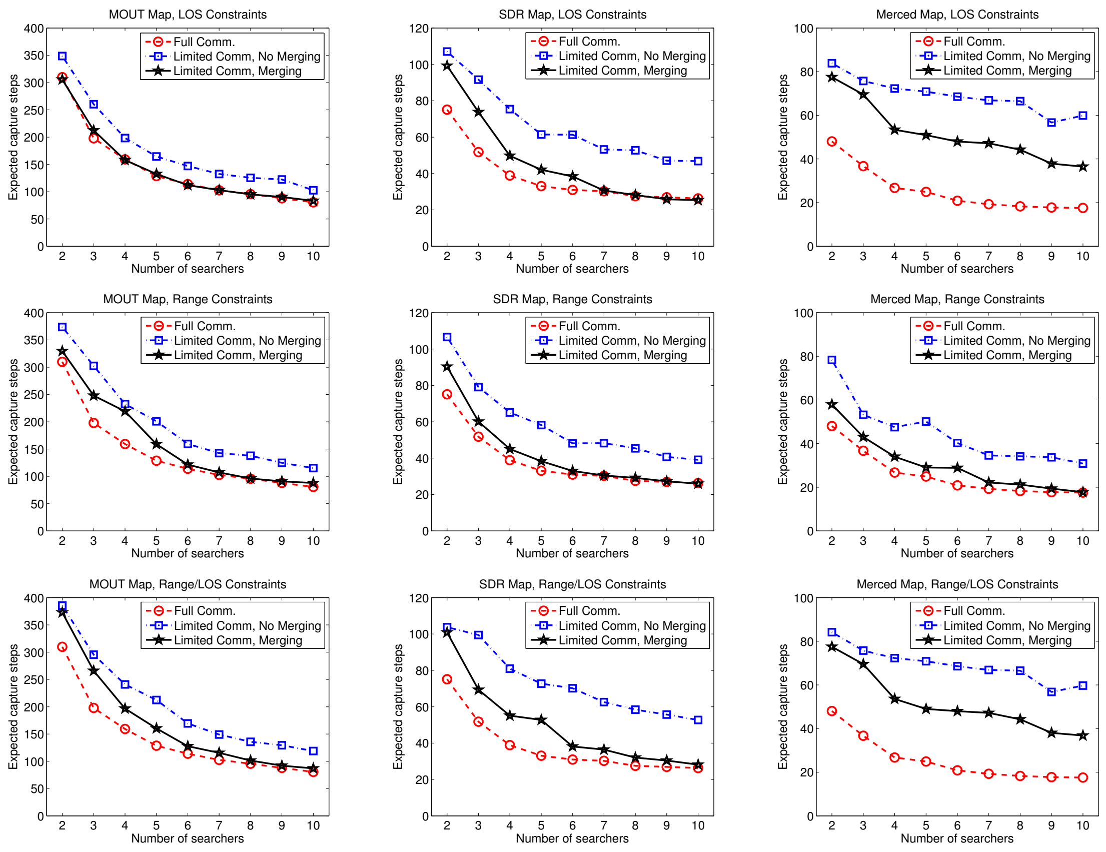


Figure 6.6: Comparison of full communication to limited communication with and without the use of a minimum merging rule. The minimum merging rule described in Equations 6.4 and 6.5 allows for searchers to share information when they regain communication, which improves the performance of the search. Results are averaged over 10,000 trials with a randomly moving target. SEMs are small and omitted.

on the Merced map. The line-of-sight constraints on this map are very restrictive, and the searchers are not able to perform the merging very often. This analysis motivates active planning to maintain connectivity such as the MIPP-PC approximation algorithm described earlier in this chapter.

6.5.5 Extension to Guaranteed Search

The minimum fusion rule can also be applied to the guaranteed search problem. In most cases, searchers performing guaranteed search must communicate only with searchers who are "freeing" them (i.e., allowing them to transition from a stationary guard to continue their schedule). Furthermore, when a cell has two possible states (cleared or contaminated), the minimum fusion rule recovers the exact distribution as long as no recontamination is allowed both in the true cleared set and in all searchers' local representations of the cleared set. However, limited communication

Table 6.1: Attained search numbers for G-GSST with continual connectivity constraints. Range constraints are shown as percentages of map diagonal.

	Cave	NSH	Museum	Merced	SDR	MOUT
No Constraints	3	3	5	7	8	9
Range (80%)	3	3	5	8	9	9
Range (60%)	3	3	5	8	9	9
Range (40%)	3	3	5	8	9	9
Range (20%)	5	5	5	8	10	11
LOS	4	-	-	-	-	11

in the presence of recontamination can cause searchers to hold different views of the cleared and dirty sets. If the positions of the other searchers are not updated, it may not be possible to determine a current representation of the cleared set, which can lead to problems executing the clearing schedule. For example, suppose that team A clears area x and then moves to area y. Further suppose that team B, who could not communicate with team A, must clear an area z that can only be cleared if area x is cleared. Team B will have no way of knowing whether or not area z can be cleared unless they communicate with team A beforehand.

Despite the problems with periodic connectivity explained above, it is easy to modify the G-GSST algorithm from Chapter 5 to handle continual connectivity. Algorithm 5.1 is modified such that any move that breaks connectivity is disallowed (treated like a recontamination move). The resulting attained search numbers with line-of-sight and range constraints are shown in Table 6.1. Line-of-sight trials were not run in the indoor environments since doorways would prevent almost any successful clearing schedule. In the six test environment shown, enforcing continual connectivity does not have a large effect on the number of searchers required.

6.6 Chapter Summary

This chapter has introduced the problem of multi-robot informative path planning with periodic connectivity (MIPP-PC). A scalable, online, and decentralized algorithm was presented that utilizes implicit coordination to solve the problem approximately. It was shown in the non-adversarial search domain that receding horizon planning with implicit coordination yields significant improvements over gradient approaches. This is due to the inability of one-step approaches to generate plans that regain connectivity and the tendency for one-step plans to fall into local maxima. Augmenting the one-step gradient with a Fiedler value heuristic eliminates some local maxima, but still does not allow for plans that temporarily lose connectivity. In addition, it was demonstrated that market-based approaches that explicitly coordinate perform only marginally better than implicit coordination, even given significantly more computational time. Thus, similar to many other search and information gathering problems, planning in the joint path space is not necessary to find good strategies with periodic connectivity. Finally, theoretical analysis was given relating the performance guarantees of implicit coordination applied to MIPP-PC to those of MIPP given various assumptions.

To complement the MIPP-PC active coordination algorithm, this chapter has presented a decentralized data fusion rule for use in search tasks with limited communication. A minimum fusion rule was introduced that recovers an approximation of the true distribution if one or more searchers are prevented from sharing information and then come into contact at a later time. Analysis was provided showing that the minimum fusion rule never underestimates the probability that the target is in a given location, and never overestimates the belief that the target has been found. Thus, the minimum fusion rule is conservative with respect to the efficient search metric. Simulated results were presented in the efficient search domain showing the benefit of incorporating the data fusion rule into search tasks with a non-adversarial target.

The analysis and results in this chapter show that implicit coordination can be applied to domains with connectivity constraints if periodic connectivity is allowed. Relaxing the continual connectivity requirement reduces the required coupling between the robots to the point where they no longer need to coordinate explicitly to plan highly informative paths. This approach can be extended to other domains if it is possible to modify the problem formulation such that the coupling between the robots' actions is reduced. In addition, the introduction of conservative data fusion rules increases the applicability of implicit coordination to domains where constant communication is not available. Through the use of such techniques, robots can perform tasks independently and then fuse their information when they come together. Taken together with the previous chapters, this chapter further confirms the effectiveness of implicit coordination when applied to multi-robot search problems that require tight coordination.

Chapter 7

Search with Subsequent Actions

THIS chapter extends the ideas already presented in this thesis by exploring the interconnection between search and a subsequent action in the context of mobile robotics. In previous chapters, the search task was considered completed when the target was found. However, in many applications, an action must be performed with the found object immediately after it is located. For example, an object must be returned to its owner, a survivor must be rescued, or an adversary must be tracked or neutralized.

The task of searching for an object and then performing some action with that object is important in many applications. One example of current interest to the robotics community is a robot assistant capable of performing worthwhile tasks around the home and office (e.g., fetching coffee, washing dirty dishes, etc.). Such a robot would need to locate objects before performing its required tasks, which makes the problem a case of search with subsequent actions. A broad range of additional applications of search with subsequent actions include autonomous Chemical Biological Radiological Nuclear (CBRN) threat identification and cleanup, mine inspection and repair, and wilderness rescue.

The following analysis provides both theoretical and experimental results for the search problem with subsequent actions. The theoretical analysis shows that some tasks allow for search and a subsequent action to be completely decoupled and solved separately, while other tasks require the problems to be analyzed together. This analysis is utilized to allow an autonomous mobile robot to perform a search and delivery task at a significantly reduced computational cost. The theoretical results are complemented by the design of a combined search/action algorithm that draws on FHPE+SA from Chapter 3. The effectiveness of this algorithm is validated by comparing it to state-of-the-art solvers, and empirical evidence is given showing that search and action can be decoupled for some useful tasks. The results in this chapter extend the principles of implicit coordination to a wider range of search tasks by including a subsequent action.[1]

7.1 Search/Action Background

Think back to the last time you lost your car keys. While looking for them, a few considerations likely crossed your mind. There were certain places in the house where the keys were more likely to be, and these places were certain distances from one another. While planning your search for your keys, you might have thought to minimize distance traveled while ensuring that you hit all the likely places that the keys might be. What may not have crossed your mind was that after finding the keys, you needed to use them to start the car. Would it be advantageous for you to search the car first because then you could immediately drive away, or would it be advantageous to search the car last to avoid backtracking to the house? Consider a similar situation where you need to find a friend at a carnival just before leaving. Might it be advantageous to search near the entrance to allow a quick exit if the friend is found? Does the fact that the friend might be moving change the answer to these questions? What if there are multiple sets of keys or friends? This chapter examines such questions in the context of mobile robotics.

A significant body of prior work in robotics has considered the *action* aspect of tasks. In other words, the robot is assumed to have at least an approximate estimate of the position of all objects of interest. However, it is often overlooked that robots must *search* for the objects (or targets) of interest before using them to perform tasks. A concrete example is a robot that refreshes coffee in an office. This robot must find coffee mugs, wash them, refill them, and return them to office workers. It is unreasonable to assume that such a robot could maintain an accurate estimate of the locations of all mugs and people in the environment without needing to search.

Fitting with this example, research in assistive robotics typically does not consider searching for an object before performing an action on it (Jain and Kempe, 2010; Srinivasa et al., 2010; Quigley et al., 2007). These systems utilize computer vision to find objects, but they are given a coarse estimate of each object's position (e.g., the table on which it rests). Thus, they do not search on the scale of an office building or home. One notable exception is the work of Roy et al. (2003) in which robots search for patients in a nursing home. However, this research falls in the category of pure search: they do not consider performing any action with the target after it is found.

Similarly, there is growing interest in developing urban search and rescue robots capable of finding survivors (and lost first responders) in disaster scenarios (Kumar et al., 2004). Work in this area typically deals with the *search* aspect of the task, but rarely is the actual rescue considered. What is ignored is that, in some cases, the specifics of the rescue (or *action*) might affect the search task. For instance, if certain areas of the building are particularly easy extraction points, it may be beneficial to

[1]The algorithms in this chapter were originally introduced by Hollinger et al. (2009a).

search these areas first. Such considerations have not seen a principled analysis in the literature.

7.2 Search/Action Problem Setup

This section formulates the combined search/action problem and shows how it can be expressed as a Partially Observable Markov Decision Process (POMDP). The formulation starts with a single searcher and single stationary target on a graph and then is extended to multiple moving searchers and physical environments. As in previous chapters, assume a graph $G = (N, E)$ is given with $|N|$ vertices and $|E|$ edges. A searcher exists at any time t at a vertex $s(t) = u \in N$, and the searcher can move to any vertex $s(t+1) = v \in N$ if an edge exists in E between u and v. Similarly, a target exists at some vertex $e(t) = u \in N$. A searcher at a given vertex can detect any target at the same vertex with some fixed probability $\Pr(\mathbf{C}|s(t) = e(t))$, where $\mathbf{C}$ is a detection (or capture) event. The searcher's goal is to locate the target and perform some subsequent action with the target. We will refer to the subsequent action as the *action task*, which is distinct from actions performed during search. The cost of the action task is dependent on the vertex at which the target was found. We can now define the searcher's objective function $J(U)$ as in Equation 7.1. Note that the objective function depends on the term τ, which is the time to complete the combined search/action task (as defined below).

$$J(U) = \sum_{t=0}^{T} \gamma^{\tau} \Pr(\mathbf{C}|U(t) = e(t)) \Pr(U(t) = e(t)), \tag{7.1}$$

where $U = [U(1), \dots, U(T)]$ are the deterministic moves of the searcher specifying its location (i.e., $s(t) = U(t)$). The search/action time $\tau = t + \alpha(U(t))$, where $\alpha(U(t))$ is the time to complete the action task if the target is found at vertex $U(t)$. The discount factor $\gamma \in (0, 1)$ is a measure of the importance of finding the target quickly. The searcher's goal is to choose $U(1), \dots, U(T)$ so as to maximize $J(U(1), \dots, U(T))$. The searcher must then complete the action task after finding the target.

For the search/action POMDP, the states are defined by the cross product of the searcher and target locations, the actions are the searchers' movements on the graph, and the observation probabilities are defined by $\Pr(\mathbf{C}|s(t) = e(t))$. The initial belief is a potentially multi-modal estimate of the location of the target at the start of the search. To complete the formulation, a set of states are added in which the searcher has control of the target. These states are reached once the searcher has captured the target, and their associated actions represent an arbitrary task. Reward is received after the action task is completed, and a negative reward can be added for each searcher step. The use of negative rewards allows a POMDP formulation with or without a discount factor.

To extend to a moving target, simply modify the transition probabilities to account for the target's motion model.[2] To extend to multiple searchers and/or targets, place K searchers on vertices $s_k(t)$ and M targets on vertices $e_m(t)$. The state and action space is now the cross product of the searchers' states and actions. This state space increases exponentially with the number of searchers and targets.

Applying the search/action POMDP to a physical workspace requires representing the workspace as a discrete graph. This can be done using any of the discretization methods used throughout this thesis (see Chapter 5 for example environments). This formulation implicitly assumes that the searchers have a sensor capable of locating a target in the same cell and that the discretized cells (corresponding to vertices) be convex and sufficiently small.

7.3 Analysis of Search with Subsequent Actions

In this section it is shown that the search tasks and action tasks can be decoupled for a single stationary target (e.g., a coffee mug) if an undiscounted reward metric is used. Conversely, it is shown that this is not the case for multiple targets, moving targets, and/or when using discounted reward. Moving targets of interest include patients in a nursing home and survivors in a search and rescue scenario.

7.3.1 Single Stationary Target

For a single searcher and stationary target located in a workspace, the searcher must find the target and then perform some subsequent action on the target. The cost of the subsequent action is dependent on the location at which the target was found. The searcher's goal is to find the target in the workspace and then complete the subsequent action. Theorem 7.1 shows that the searcher can achieve the optimal strategy for the search/action task by solving the search and action tasks separately.

Theorem 7.1 *Adding a subsequent action cost does not change the optimal search strategy w.r.t. expected cost for a single stationary target.*

Proof Let $|N| = 2$, and label these nodes n_1, n_2. The probability of the target being at each node is p_1, p_2. A fixed (or expected) action cost is associated with each node, a_1, a_2. A cost to reach each node from the start is denoted as d_1, d_2, and a cost between them is denoted as d_{12}. The searcher can choose to visit n_1 first or to visit n_2 first. The equations below show the expected cost for these two cases.

For search:

$$\text{Go to } n_1 \text{ first: } C_1 = p_1 d_1 + p_2(d_1 + d_{12})$$
$$\text{Go to } n_2 \text{ first: } C_2 = p_2 d_2 + p_1(d_2 + d_{12})$$

For search/action:

[2]Note that this formulation requires that the target's motion model be Markovian.

$$\text{Go to } n_1 \text{ first: } C_{a1} = p_1(d_1 + a_1) + p_2(d_1 + d_{12} + a_2)$$
$$\text{Go to } n_2 \text{ first: } C_{a2} = p_2(d_2 + a_2) + p_1(d_2 + d_{12} + a_1)$$

Subtracting we get:

$$C_{a1} - C_1 = p_1a_1 + p_2a_2$$

We also see that:

$$C_{a2} - C_2 = p_2a_2 + p_1a_1$$

Therefore:

$$C_{a2} - C_2 = C_{a1} - C_1$$

Generalizing to more than two possible locations, we see that any search strategy $\mathcal{S}$ and corresponding search/action strategy $\mathcal{A}$ visiting N locations in the sequence $\{1, \ldots, N\}$ will take the following form:

$$C_{\mathcal{S}} = p_1d_1 + \ldots + p_N(d_1 + d_{12} + \ldots + d_{N-1\ N})$$
$$C_{\mathcal{A}} = p_1(d_1 + a_1) + \ldots + p_N(d_1 + d_{12} + \ldots + d_{N-1\ N} + a_N)$$

$$C_{\mathcal{A}} - C_{\mathcal{S}} = \sum_{i=1}^{N} p_ia_i \tag{7.2}$$

The expected cost of completing the subsequent action is given by Equation 7.2 and is independent of the search strategy. ∎

Theorem 7.1 applies to the *undiscounted* POMDP formulation by using the shortest path distances between nodes u and v as the d_{uv}. Thus, the search POMDP and action task (PO)MDP can be solved sequentially without sacrificing optimality. This is a potentially large gain because it avoids a large expansion of the state space. Additionally, this theorem demonstrates that the subsequent action cost has no effect on the optimal search strategy, which means that knowledge of the subsequent action is unnecessary during search.

One important caveat, however, is that Theorem 7.1 is only true if the POMDP reward (inversely proportional to the cost) is not discounted. Adding a discount factor modifies the reward in such a way that the expected subsequent action cost is dependent on the search strategy. The intuition is that the discount factor allows the search strategy to modify the subsequent action cost of a location by adjusting the time at which it is visited.

For example, given two possible locations n_1 and n_2 that are one step apart and have an equal probability of containing the target. Let the reward for completing the action task (after finding the target) be $R = 1$, let the discount factor $\gamma = 0.9$, let the action costs be $a_1 = 100$ and $a_2 = 1$ if the target is found in n_1 or n_2 respectively.

Searching n_1 first yields expected reward $R_1 = 0.5 * 0.9^{100} + 0.5 * 0.9^2 = 0.405$. Searching n_2 first yields expected reward $R_2 = 0.5 * 0.9^1 + 0.5 * 0.9^{101} = 0.450$. If the action costs were uniform or the discount factor were not included, the expected reward would be indifferent between the two strategies. However, adding the discount factor makes visiting n_2 first preferable.

7.3.2 Multiple Stationary Targets

Now let there be multiple stationary targets that a searcher must find before completing the subsequent action task. An example would be if a service robot needed to collect several mugs and then take them to the dishwasher. It is also assumed that the searcher can carry more than one target, so it may continue to search after locating target. If multiple targets are considered, and the action cost is dependent on location, Theorem 7.1 does not hold. To see why, examine the expected reward for the search/action task with two targets. Let p_{ij} be the probability of finding target i in location j.

For search:

$$C_1 = p_{11}p_{21}d_1 + (p_{12}p_{22} + p_{11}p_{22} + p_{21}p_{12})(d_1 + d_{12})$$
$$C_2 = p_{12}p_{22}d_2 + (p_{11}p_{21} + p_{12}p_{21} + p_{22}p_{11})(d_2 + d_{12})$$

For search/action:

$$C_{a1} = p_{11}p_{21}(d_1 + a_1) + (p_{12}p_{22} + p_{11}p_{22} + p_{21}p_{12})(d_1 + d_{12} + a_2)$$
$$C_{a2} = p_{12}p_{22}(d_2 + a_2) + (p_{11}p_{21} + p_{12}p_{21} + p_{22}p_{11})(d_2 + d_{12} + a_1)$$

Subtracting, we see that the ratio $C_{a2} - C_2 \neq C_{a1} - C_1$ for the multiple target case. Thus, the subsequent action cost can modify the optimal search strategy. Intuitively, the reason is that, in the multiple target case, the search strategy can affect the likelihood of finishing the search at a given location. The search is most likely to end at the last point visited because targets have been gathered from all other points. If a location has a very low subsequent action cost, leaving it for last may be a desirable search/action strategy even if it is not a desirable search strategy.

7.3.3 Moving Targets

For the moving target case where action cost is dependent on location, Theorem 7.1 also does not hold. The probability of a moving target being at a given location is dependent on when that point is visited (i.e., p_i is dependent on t, yielding $p_i(t)$). The dependence of the probability of capture on time breaks the equality so that $C_{a2} - C_2 \neq C_{a1} - C_1$. For example, a target may be moving toward a location with a very low subsequent action cost. It may be advantageous for the searcher to move to that location in the search/action case even if the probability of the target being there is low compared to alternative locations.

7.3.4 Implementation on a Mobile Robot

To demonstrate the importance of decoupling search and action, tests were run on a mobile manipulator platform consisting of a Barrett anthropomorphic arm and hand mounted on a Segway mobile base. The platform localizes itself using Adaptive Monte-Carlo localization with a laser rangefinder and runs using the Open-Rave (Diankov and Kuffner, 2008) and Player (Gerkey et al., 2003) software. It carries a miniature camera, which it uses to identify coffee mugs for grasping with the arm/hand (Srinivasa et al., 2010). The system is shown in Figure 7.1.

The robot was given three waypoints in the environment that may contain mugs. The optimal path between the waypoints (w.r.t. navigation cost) was computed using an exhaustive search. Since the navigation cost is undiscounted, Theorem 7.1 applies, and we only need to consider the search costs. The application of Theorem 7.1 avoids calculating action costs for all possible locations, which would require examining the planning, manipulation, and localization necessary for each task. The robot proceeded to search these waypoints by moving to them and scanning them with the camera. Upon finding a mug, the robot would pick it up with the arm/hand and take it back to the sink.[3] The path of the robot through the office is shown in Figure 7.1.

The implementation of this search/action task on the physical mobile manipulator demonstrates the gains from decoupling search and action for a single stationary target. Determining the subsequent action cost of picking up a mug and taking it to a specified location requires the costly instantiation of navigation, object recognition, and manipulation components. The application of Theorem 7.1 allows for the robot to solve the search task without computing the subsequent action cost for all locations, which leads to significant computational gains even in this small experiment.

7.4 Search/Action Algorithm

This section combines the theoretical analysis above with the FHPE+SA algorithm from Chapter 3 to design an approximate algorithm for the search/action problem. For a single stationary target with undiscounted reward, Theorem 7.1 has shown that the search and subsequent action can be solved separately. Thus, search algorithms can be used without modification in this case.

In contrast, for discounted reward cases, it may be advantageous to consider the subsequent action cost as in Equation 7.1. The finite-horizon path enumeration (FHPE) algorithm was shown in Chapter 3 to provide high-quality, scalable results for discounted search POMDPs. The algorithm examines all possible paths to a fixed horizon and then replans on a receding horizon. FHPE is particularly well-suited to problems where maximizing short-term reward does not conflict with maximizing

[3]Multimedia extensions (see Appendix A) show a video of the mobile manipulator performing the search and subsequent action.

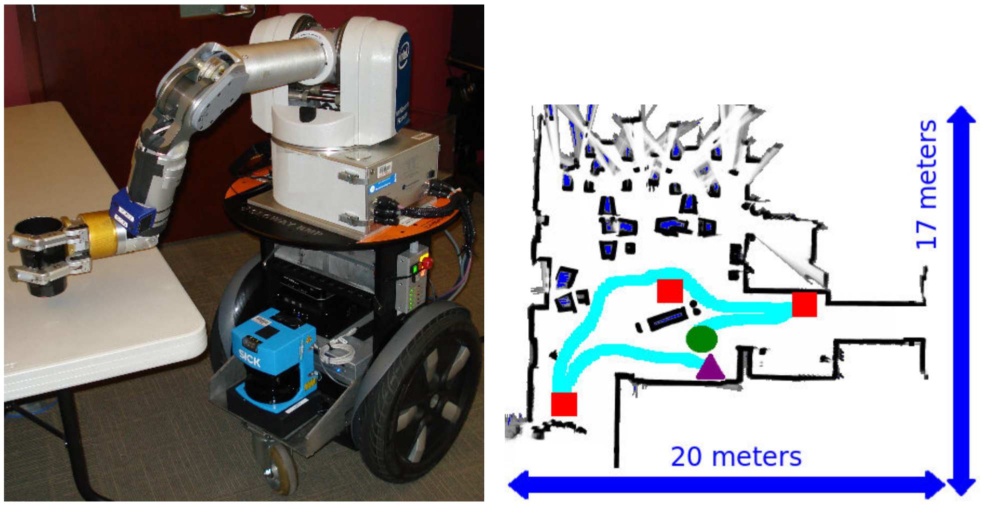


Figure 7.1: Left: Mobile manipulator: Segway RMP200 base with Barrett WAM arm and hand. The robot uses a wrist camera to recognize objects and a SICK laser rangefinder to localize itself in the environment. Right: Map of a kitchen area within the Intel labs used for implementation of search/action on mobile manipulator. Circle shows starting robot location, squares show possible locations of coffee mugs, and triangle shows the location to which the robot must move the mug. An example robot path is shown in cyan (light grey); the mug was found in the bottom left location in this trial.

Algorithm 7.1 FHPE for search/action problems

```
1: Input: Single-searcher search/action problem
2: for All paths $U$ to horizon $d$ do
3:     Calculate $J(U)$ as in Equation 7.1
4: end for
5: $U^* \leftarrow \arg\max_U J(U)$
6: while Target not found do
7:     Execute $U^*$ replanning as needed
8: end while
9: Perform action task with found target
```

long-term reward. This is often the case during search/action tasks because searchers will usually search nearby locations before searching further away.

Algorithm 7.1 shows an application of FHPE to the discounted search/action problem. Depending on the exact instance, the reward function $J(U)$ can be substituted as appropriate. For both moving and stationary targets, the reward function is the time-truncated version of Equation 7.1, which includes the cost of the subsequent action.

Algorithm 7.2 Implicit coordination for search/action problems

```
1: Input: Multi-searcher search/action problem
2: % $V \subseteq N'$ is the set of nodes visited by searchers
3: % A node in $N'$ is a time-stamped node of $N$
4: $V \leftarrow \emptyset$
5: for all searchers $k$ do
6:    % $U_k \subset N'$ is a feasible path for searcher $k$
7:    % Finding this arg max solves the search/action for searcher $k$
8:    $U_k^* \leftarrow \arg\max_{U_k} J(V \cup U_k)$
9:    $V \leftarrow V \cup U_k^*$
10: end for
11: while target not found do
12:    Execute $U_k^*$ for all searchers $k$ replanning as needed
13: end while
14: Perform action task with found target
```

7.4.1 Multiple Searchers

The finite-horizon approximation algorithm can be extended to multiple searchers using implicit coordination. The multi-searcher reward function is shown in Equation 7.3.

$$J(U) = \sum_{t=0}^{T} \gamma^{\tau} \Pr(\mathbf{C} | \exists k : U_k(t) = e(t)) \Pr(\exists k : U_k(t) = e(t)), \tag{7.3}$$

where $U(t) = [U_1(t), \ldots, U_K(t)]$ is now a K-dimensional control vector specifying the location of each searcher (i.e., $s_k(t) = U_k(t)$). The search/action time $\tau = t + \alpha\,(U_1(t), \ldots, U_K(t))$, where $\alpha(U_1(t), \ldots, U_K(t))$ is the expected time to complete the action task if the target is found on each of vertices $U_1(t), \ldots, U_K(t)$ weighted by the probability that the target is at each vertex.

Algorithm 7.2 gives an implicit coordination solution to the search/action problem. The searchers sequentially allocate their paths through the environment, and then simultaneously execute these paths. The algorithms run on the time-unfolded graph N' (see Chapter 3), which allows searchers to revisit locations and for more than one searcher to be in the same location simultaneously. Sequential allocation provides linear scalability in the number of searchers, and gives a constant factor approximation guarantee for nondecreasing, submodular objective functions. The search/action objective function $J(U)$ is both nondecreasing and submodular (proof is omitted but is a straightforward corollary of the theoretical analysis in Chapter 3), which leads to a bounded approximation algorithm for the multi-searcher search/action problem.

7.5 Search/Action Experiments

7.5.1 Simulated Environments

To test the search/action algorithm, a simulated scenario was set up requiring both a search and subsequent action. One or more searchers move around an indoor environment with omnidirectional line-of-sight sensors. The searchers must locate a stationary or moving target and then take the target back to the starting location. The searchers use a *discounted* reward metric, which means that Theorem 7.1 does not apply in either the stationary or moving cases. Simulations were run in the office and museum environments (see Chapter 3). The starting location of the target is initialized randomly, and the target's movement (if applicable) is a random walk. The simulator runs in C++ in Linux on a 3.0 GHz P4 with 2 GB RAM.

The proposed algorithm was compared to two state-of-the-art POMDP solvers: HSVI2 and SARSOP. Since these solvers are optimized with discounted reward in mind, the experiments are limited to this case (with arbitrarily set $\gamma = 0.95$). The POMDP solvers were given two minutes of solving time for each instance. Unlike in Chapter 3, the additional complexity of the action cost prevented the solvers from converging to a near-optimal solution. After two minutes, the bounds on solution quality stopped improving, but they were still far from being fully converged. Since FHPE runs online with a receding horizon, it did not need to precompute a solution. Table 7.1 shows that FHPE provides solutions competitive with the POMDP solvers in these environments, with the exception of the case of a moving target in the museum. The higher complexity of the museum map requires a significant lookahead, which improves the benefit of using the POMDP solver. The longer lookahead of the POMDP solver allows the searchers to more effectively "trap" the moving target in areas with low action costs. It is important to note that the number of states in these POMDPs (approximately 5000) is at the frontier of what general POMDP solvers can handle. FHPE, on the other hand, is scalable to much larger environments.

Table 7.1 also compares two versions of FHPE applied to the search/action problem. The first version decouples search and action by excluding the subsequent action costs in the calculation of the search strategy. The full version of FHPE includes subsequent action costs as in Equation 7.1. The results show that, in these environments, taking into account the subsequent action cost yields only a small improvement in the final reward. This is somewhat contrary to the theoretical analysis in Section 7.3, which shows how search and action *cannot* be provably decoupled with a discounted reward metric. However, the results show that one can "get away" with decoupling search and action in these instances. This is likely due to the relatively small differences between the subsequent action costs at each location in these scenarios.

Trials with multiple searchers were also run using FHPE+SA. The searchers move through the environment until the target is found, and the searcher that locates the target then takes it back to the starting cell. Figure 7.2 shows the results from the

Table 7.1: SEARCH/ACTION reward comparison (higher is better) of HSVI2 (Smith, 2007), SARSOP (Kurniawati et al., 2008), and FHPE (with and without including action cost) for a single searcher in two environments. Averages are over 200 trials, and errors are one SEM.

Environment (target)	HSVI2	SARSOP	FHPE (w/o action cost)	FHPE (with action cost)
Office (stationary)	17.3 ± 3.3	21.3 ± 3.3	21.3 ± 1.9	22.8 ± 1.8
Museum (stationary)	14.2 ± 2.5	8.2 ± 3.0	17.5 ± 1.5	18.5 ± 1.7
Office (moving)	46.2 ± 3.8	47.3 ± 2.7	47.2 ± 1.6	47.7 ± 1.5
Museum (moving)	27.5 ± 3.6	29.9 ± 3.4	21.6 ± 1.5	22.5 ± 1.5

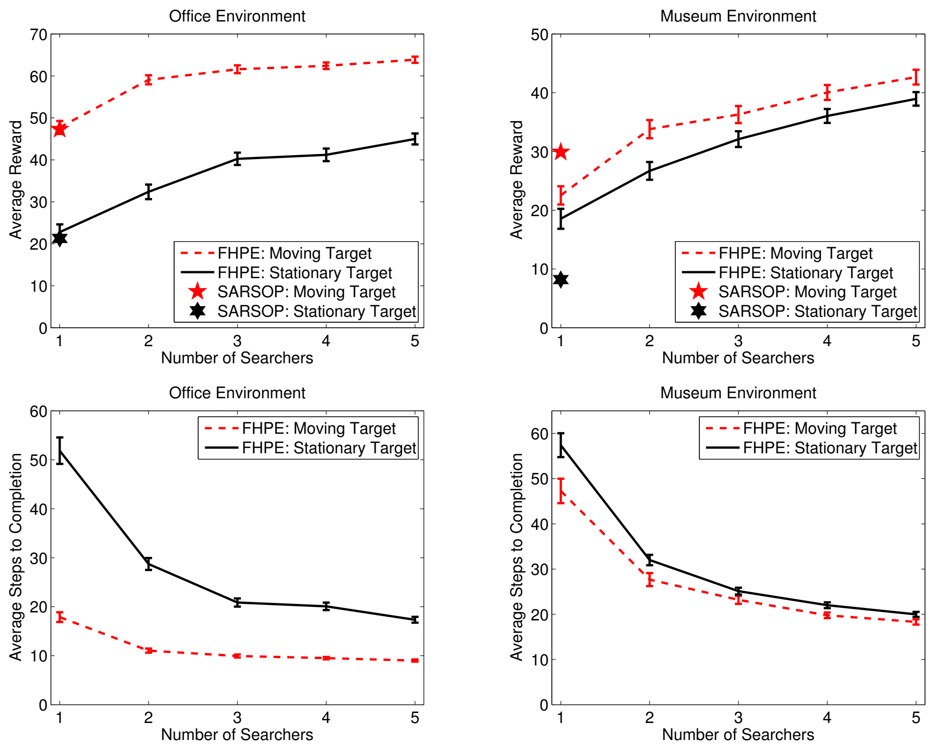


Figure 7.2: SEARCH/ACTION results using FHPE and implicit coordination with multiple searchers. Averages are over 200 trials, and error bars are one SEM. SARSOP, a general POMDP solver, was unable to fit the two searcher instance in memory. Top graphs show average reward (higher is better), and bottom graphs show average steps to complete the search/action task (lower is better).

multi-searcher trials. The linear scalability of FHPE+SA easily handles five searchers in these environments. Neither of the general POMDP solvers were able to fit the two searcher problem in memory.

7.6 Chapter Summary

This chapter introduced the connection between search and subsequent action to mobile robotics. The problem of locating a target and then performing an action with that target was examined, and the relevant applications were explored. Theoretical analysis was presented showing that the search/action problem with a single stationary target can be solved by considering the search and action components separately. For discounted reward, multiple targets, and moving targets, on the other hand, it can be beneficial to consider the action component before performing the search. Drawing on this theoretical analysis, approximate algorithms were designed for the search/action task with both stationary and moving targets.

The performance of the proposed algorithm was validated through simulated experiments. The simulated results showed that solving search and action separately has only a small affect on solution quality if the action costs are not highly disparate across the environment, which is often the case in search and retrieval tasks. It was also shown that the proposed algorithm is competitive with general POMDP solvers for moderately sized problem instances. General POMDP solvers require resources beyond computational limits with multiple agents and in large environments. In contrast, the proposed algorithm uses a receding-horizon technique to remain scalable in large environments, and it utilizes implicit coordination to achieve linear scalability in the number of searchers.

This chapter showed that a simple sequential planning approach yields good performance even when a subsequent action is required. Based on the analysis in previous chapters, this result is somewhat expected, since the addition of a subsequent action does not increase the coupling between the searchers and does not remove the diminishing returns property from the objective function. Even though the planning space is larger, the properties of the underlying objective remain the same. In many of the cases explored in this chapter, it is not even necessary to consider the subsequent action to achieve good performance. Thus, the algorithms from the efficient search domain can be applied without the need for introducing an augmented environment representation (as in Chapter 4) or reducing the coupling (as in Chapter 6).

Chapter 8

Integrative Example

THIS chapter integrates the ideas in this thesis to solve the search and secure problem from the Multi Autonomous Ground-robotic International Challenge (MAGIC). The competition requires a team of heterogeneous robots to locate, classify, and secure a number of targets in an indoor/outdoor environment. The algorithms proposed in earlier chapters are extended to solve the coordination aspects of the challenge by providing guaranteed clearing strategies (i.e., strategies that guarantee coming into contact with any adversarial target). The proposed method allows for repair of the clearing schedule after robot failure, as well as a fall-back efficient search strategy if clearing is no longer possible. A scenario taken directly from the competition is analyzed, and it is demonstrated in simulation that extensions of the algorithms in this thesis provide high mission success rates.

The key idea behind the proposed coordination strategy is for the robot team to start by generating a guaranteed schedule that ensures mission completion. If a failure occurs during execution, the robots replan to determine if they can still execute a guaranteed schedule by repairing the plan. If no such guaranteed schedule is found, the robots resort to efficient search to complete the mission. If all robots are disabled, then the mission is considered a failure, an outcome that should be avoided as much as possible.

The MAGIC competition requires autonomous coordination of a heterogeneous team of ground robots to locate, classify, and secure all simulated threats in a complex indoor/outdoor urban environment. To apply the techniques in this thesis to the competition, a number of improvements were necessary. These improvements include: the coordination of heterogenous "disruptor" robots to secure targets, the extension of G-GSST to allow for dynamic replanning, and the integration of a lattice planner for collision avoidance during planning and execution. The MAGIC results in this chapter demonstrate the successful application of principles discussed in this thesis to a problem of significant interest to the robotics community.

8.1 MAGIC Competition Rules

The MAGIC 2010 guidelines (Defence, Science and Technology Organisation, Land Operations Division, 2009) describe the following rules (paraphrased):

1. To complete each challenge phase a team must: a) explore and map the entire phase area; and b) correctly locate, classify, recognize and secure all targets.
2. The team of UGVs is divided into sensor UGVs (for mapping and detection) and disruptor UGVs (to secure detected targets). The team must have a ratio of at least two sensor UGVs to disruptor UGVs.
3. Mobile targets will move at a maximum velocity of 6 km/hour. They may stop, turn, about face, reverse or continue maneuvering.
4. Mobile targets have detection and lethality zones of 10 m diameter (5 m radius), subject to building occlusion. Any UGV entering the lethality zone of a mobile target will be deemed to have been detected and damaged.
5. Mobile targets must be secured to complete the challenge. To secure a mobile target, a disruptor and sensor UGV must simultaneously view it. The target must then be continuously viewed and tracked by both UGVs for a period of 15 seconds.[1]
6. Static targets have an activation zone of 5 m diameter (2.5 m radius), subject to building occlusion. Any UGV entering the activation zone of a static target will cause it to detonate. Static targets have a lethality zone of 20 m diameter (10 m radius), subject to building occlusion. If detonated, any UGV, target or non-combatant within the lethality zone will be deemed to have been damaged.
7. To secure a static target, one or more sensor UGVs must locate and classify it, and the team leader must communicate this information to judges. A disruptor UGV must then approach the target to within 2 m of its activation zone (within 4.5 m of the target) and view it for at least 30 sec.
8. During one or more phases of the challenge, at least one UGV will be lost to enemy "sniper" action.

Several rules were simplified to decouple the problem and highlight the contributions of this thesis. The following rules were modified as noted:

1. The environment is assumed to be known a priori. If this is not the case, the application of replanning methods from Chapter 5 is straightforward.

[1]The maximum sensor range of the UGVs is not specified in the competition rules and is determined by the specifics of the detection system. In this chapter, a maximum sensor range of 50 m is assumed.

2. Non-combatants are not present in the environment. The presence of non-combatants would primarily affect the actions of the disruptors once a target is acquired. Thus, they can be decoupled from the search task.

3. All-to-all communication is allowed between the vehicles. Active and/or passive connectivity management techniques from Chapter 6 could be applied if this assumption were violated.

4. Random failures due to "sniper" action and other failures are not examined. These would decrease the likelihood of mission success, but they would obfuscate the analysis of failures due to target proximity.

8.2 Search and Secure Scenario

The Phase 2 Scenario (see Figure 8.1) from the MAGIC information document (Defence, Science and Technology Organisation, Land Operations Division, 2009) was implemented in the Player/Stage simulator. The 200 m $\times$ 200 m environment contains five targets, three stationary and two mobile. The mobile targets patrol around the building as shown in the diagram at a speed of 4 km/hr. Any searcher or disruptor that comes within 5 m of a mobile target is disabled and deemed unable to function (i.e., it cannot move, detect, or secure targets). If any searcher or disrupter comes within 2.5 m of a stationary target, the target detonates and disables all robots within 10 m.

The searchers start in the northwest corner of the map and must locate and secure all five targets to successfully complete the mission. The searcher vehicles are modeled as having a maximum speed of 10 km/hr, which is the imposed speed limit from the competition. The vehicles are assumed to be car-like vehicles with the appropriate non-holonomic constraints. If two vehicles collide, they are also considered to be disabled; however, this event rarely occurs due to low-level collision avoidance. The collision avoidance is provided by a lattice planner, which was implemented specifically for this domain and based on work by Likhachev and Ferguson (2009). Each robot's lattice contains obstacle regions for the current positions of the other robots. The planner is used online to avoid inter-robot collisions.

8.3 Coordination Strategy

As mentioned above, the team starts with a guaranteed schedule and attempts to repair this schedule if failures occur. If a guaranteed schedule is no longer possible, the team resorts to an efficient search schedule. A modification of the G-GSST algorithm was used to generate guaranteed schedules for the search and secure scenario. First, the environment was discretized into 32 cells (shown in Figure 8.2) using the region growing technique described in Chapter 5. The largest dimension of the cells was

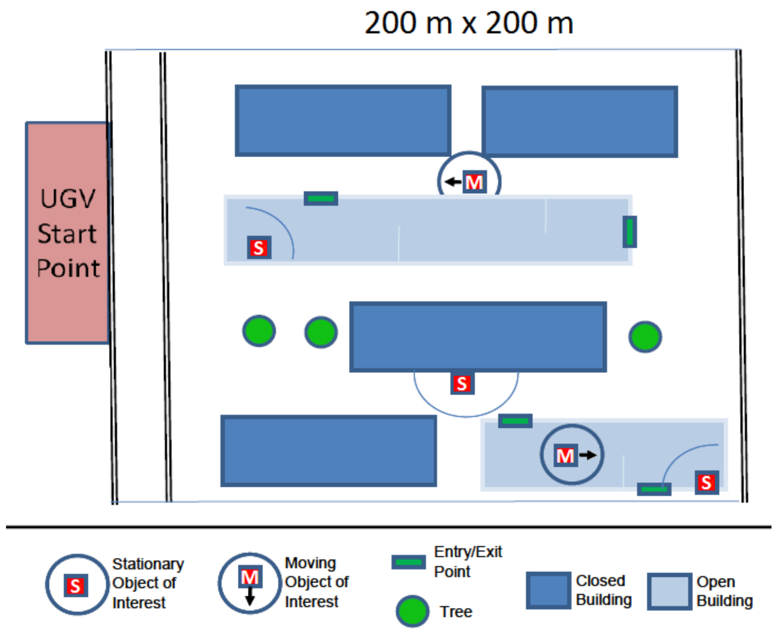


Figure 8.1: Phase 2 environment from the MAGIC competition guidelines (Defence, Science and Technology Organisation, Land Operations Division, 2009). A heterogeneous team of UGVs must locate and secure all mobile and stationary objects of interest in the environment.

limited to 50 m to correspond to the UGV sensor range. Next, G-GSST was run to find a clearing schedule on the discretized map.

After a clearing schedule was found, the locations of the disruptors were determined at each step using a sequential minimization of the average distance of each searcher to a disruptor. Each disruptor determines what one-step move it should make to minimize the metric and then shares that information with the other disruptors. The other disruptors consider the move fixed when planning their own moves. Note that this disruptor coordination strategy is an implicitly coordinated solution to disruptor allocation that is decoupled from the generation of the search schedule.

When a target is located during the search, the searcher calls for the nearest disruptor to come secure the target. If that searcher is not available, the next furthest disruptor is called until an available disruptor is found. The disruptor moves to the current location of the searcher, rather than the centroid of the cell. Thus, the discretization can be coarser than the necessary detection radius and still allow for target securing. After the target is secured, the clearing schedule continues.

When robots come too close to a target, they are disabled, and the remaining team replans to determine if a clearing schedule can still be executed from their current locations. If a clearing schedule is not found, the team executes FHPE+SA to maximize the efficient search objective function (i.e., the discounted probability

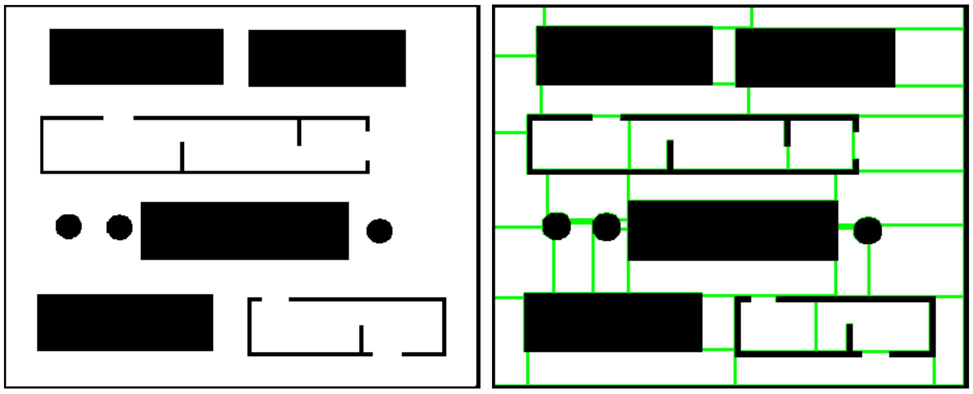


Figure 8.2: Polygonal representation of MAGIC environment (left) and discretization using a region growing technique (right).

of locating a slowly moving random target). This strategy is then executed until either all targets are secured (mission success) or all disruptors are disabled (mission failure).

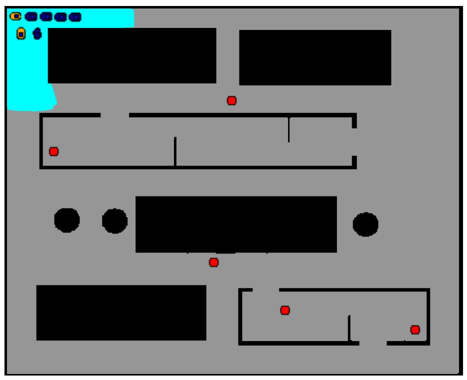
8.4 Simulated Missions

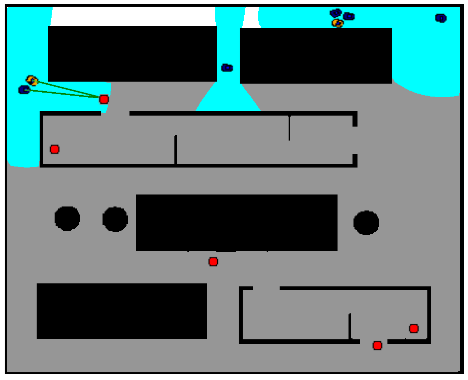
Figure 8.3 shows snapshots of the Player/Stage simulator as a seven robot team (two disruptor UGVs and five sensor UGVs) successfully locates and secures all targets in the environment. A number of simulated trials were run to determine the robustness of the proposed solution. Table 8.1 gives a summary of these robustness results.

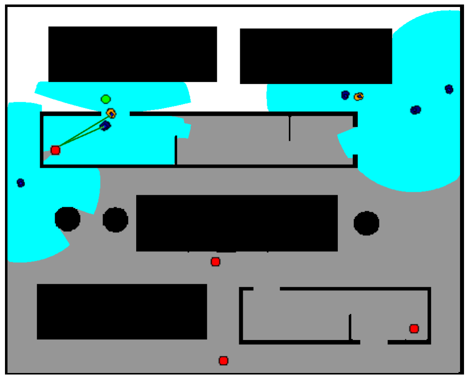
The results in Table 8.1 show completion times and percentages for different failure modes using a varying number of robots. The proposed coordination strategy achieves as high as a 97.5% success rate for mission completion with 6 searchers and 3 disruptors. Failures occurred in approximately 25% of trials, and the majority of failures eventually led to mission success using one of the two recovery strategies. The average completion times show that completion without failure was faster than completion with a soft failure, and completion with a hard failure had a high variance (ranging from fast to slow completions).[2]

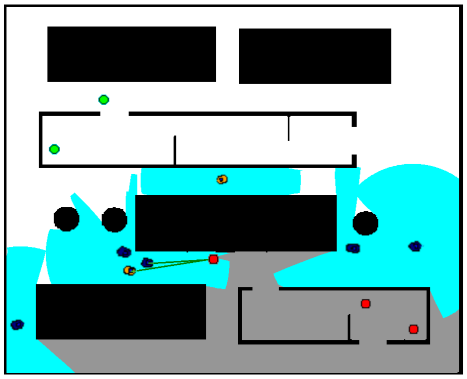
In addition, Table 8.1 shows results where one searcher is allocated to optimize a random target model using FHPE+SA while the other five searchers execute a clearing schedule (a similar allocation to the experiments shown in Chapter 5). The reallocation of an efficient searcher leads to a reduction in the mean completion time, but a large number of recoverable failures occur when the efficient searcher is disabled. Based on these results, the user may be tempted to run an efficient strategy using FHPE+SA that does not guarantee clearing. However, it is important to note that

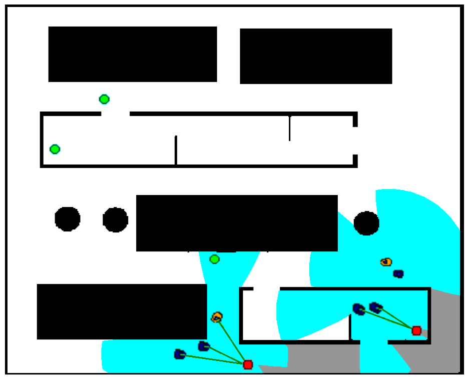
[2]Multimedia extensions (see Appendix A) show playback from the Player/Stage simulation for successful runs, including soft and hard failures.


(a) 0 sec: the searchers start in the north-west corner of the map.


(b) 2 min 24 sec: the first mobile target is secured.


(c) 5 min 25 sec: the first stationary target is secured.


(d) 9 min 54 sec: the second stationary target is secured.


(e) 13 min 15 sec: the second mobile target is secured.

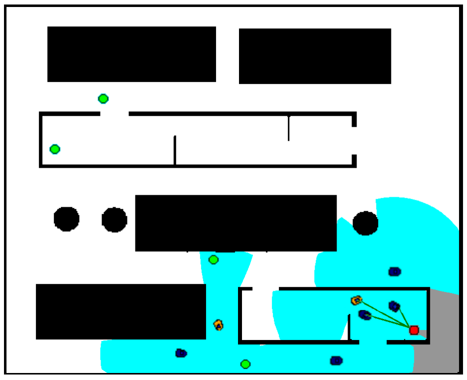

(f) 14 min 18 sec: the final target is secured.

Figure 8.3: Snapshots of Player/Stage simulation of robots executing a MAGIC search and secure task. Dark areas may contain targets; light areas are cleared. Searcher UGVs are shown in blue, disruptor UGVs in yellow, and targets in red.

Table 8.1: Robustness trials for MAGIC 2010 scenario. The cases with no failures and soft failures result in a guaranteed clearing schedule. The hard failures result in a completion without a clearing guarantee. Averages are over 200 trials; errors are one standard deviation.

5 Guaranteed Searchers, 2 Disrupt.	Percentage	Average Completion Time
Completion (no failure)	72.5%	687.41 ± 39.48 sec
Completion (soft failure)	8.0%	755.69 ± 57.75 sec
Completion (hard failure)	9.5%	854.42 ± 313.84 sec
Unable to complete	10.0%	-
6 Guaranteed Searchers, 2 Disrupt.	Percentage	Average Completion Time
Completion (no failure)	76.5%	643.44 ± 43.83 sec
Completion (soft failure)	9.0%	706.44 ± 125.93 sec
Completion (hard failure)	7.5%	885.07 ± 327.73 sec
Unable to complete	7.0%	-
6 Guaranteed Searchers, 3 Disrupt.	Percentage	Average Completion Time
Completion (no failure)	75.5%	634.86 ± 43.27 sec
Completion (soft failure)	14.0%	644.39 ± 53.91 sec
Completion (hard failure)	8.0%	698.06 ± 180.06 sec
Unable to complete	2.5%	-
5 Guaranteed, 1 Efficient, 2 Disrupt.	Percentage	Average Completion Time
Completion (no failure)	44.0%	503.60 ± 94.96 sec
Completion (soft failure)	32.0%	705.16 ± 77.55 sec
Completion (hard failure)	14.5%	821.28 ± 403.01 sec
Unable to complete	9.5%	-

few of the strategies that resorted to FHPE+SA led to mission success. In addition, running FHPE+SA from the start results in a mission failure in this scenario for all three varying robot numbers. The high failure rate of FHPE+SA likely results from the tendency for searchers to spread out, which taxes the disruptors to respond quickly. Thus, the efficient strategy is a "more risky" strategy than the clearing strategy. These results further confirm the trends in Chapter 5.

8.5 Chapter Summary

This chapter presented further validation of the techniques proposed in this thesis using an environment from the MAGIC 2010 competition. A scenario from the competition announcement was analyzed in the Player/Stage simulator. It was shown that modifications of G-GSST (Chapter 5) and FHPE+SA (Chapter 3) provide robust and effective solutions to many of the coordination problems posed in the MAGIC competition. The algorithms were also extended to allow for the coordination of a heterogenous team of robots preforming different, but interrelated, tasks of searching and securing.

The proposed solution to the MAGIC problem allows for search and secure with a heterogeneous team of searchers and disruptors, and was able to achieve mission success as much as 96% of the time. In addition, failure modes were explored that allowed for recovery of the clearing schedule and resorted to efficient search as necessary. The results in this chapter demonstrate that riskier strategies for finding average-case targets tend to fail more often than more conservative strategies for finding worst-case targets. This trend coincides with insights from Chapter 5 as well as ongoing work in the search literature (Strom et al., 2010). Simulated experiments allowed for this analysis, which would not have been possible based on the few robot trials from the competition itself.

In addition to the validation of the techniques in this thesis on a problem of significant interest to the robotics community, the results in this chapter demonstrate the applicability of implicit coordination on a team of heterogeneous agents. The searchers perform the clearing task, while the disruptors minimize their distance to the searchers and move to secure targets when necessary. The task of minimizing average distance has a similar property of diminishing returns as discussed in the efficient search domain (Chapter 3). Thus, a simple sequential approach allows for effective disruptor strategies. The results in this chapter also highlight the modular nature of implicit coordination (as discussed in Chapter 5). Each robot can perform its own distinct task while utilizing the information about the plans of its teammates. The successful integration and application of implicitly coordinated solutions to a problem with a heterogeneous team further highlights the wide range of problems that can be solved using the approach.

Chapter 9

Conclusions and Future Directions

9.1 Conclusions

The algorithms proposed in this thesis utilize implicit coordination, where each robot plans its own path and shares that information with the rest of the team in a decentralized manner. Implicit coordination prevents the exponential increase in computation required to coordinate the joint actions of the entire team, and it avoids the computation and communication overhead of running auctions or team formation algorithms. Implicit coordination, together with an informed environment representation, allows for the execution of complex search tasks with high-performing results. The proposed architecture was shown to outperform a number of state-of-the-art algorithms in the efficient, guaranteed, and constrained search domains. In addition, validations were presented using scenarios from the MAGIC 2010 competition, an international robotics competition that is currently underway.

Beyond search, the analysis in this thesis provides insight into the properties that make certain multi-robot problems *tightly coupled*. Tight coupling, as discussed extensively in the market-based literature (Kalra, 2006), requires extensive interaction between the searchers to achieve good performance. While all search tasks may seem tightly coupled at first, further analysis shows that efficient search (i.e., the optimization of a non-adversarial target model) does not require tight coordination. This property can be formalized using the diminishing returns property of submodularity as discussed in Chapter 3. When the cost function of a multi-robot coordination problem demonstrates this property of diminishing returns, planning algorithms that do not require extensive interaction (e.g., implicit coordination) can be expected to perform well.

In contrast, problem domains that do not exhibit the quality of diminishing returns (e.g., search for an adversarial target), may require additional interaction to achieve good performance. The poor performance of receding-horizon greedy clearing in the guaranteed search domain (see Chapter 4) demonstrates this tendency. In these domains, it is necessary to develop an informed environment representation, such as a spanning tree, to guide the use of implicitly coordinated methods. The development of such representations, however, is domain specific and may not be possible for all tightly coupled problems. The addition of inter-robot constraints, such as connectivity, also increases the coupling between robots. In many cases, such inter-robot constraints also break the diminishing returns quality. However, techniques that loosen the coupling, such as allowing for periodic connectivity as proposed in Chapter 6, open these domains to implicitly coordinated techniques.

Another interesting observation is that the coupling between robots in a particular domain is closely related to the existence of performance guarantees for scalable algorithms in that domain. In the efficient search domain, it is possible to achieve a constant factor approximation using algorithms that are linearly scalable in the number of robots. However, in the guaranteed and constrained search domains, no approximation guarantee is possible unless $P = NP$. Furthermore, a loose performance guarantee can be gained in the constrained search domain by introducing periodic connectivity, which clearly reduces the coupling between robots. Thus, the analysis of formal hardness leads to a principled method for examining multi-robot problems and determining whether they are *difficult* or *easy* to solve.

9.2 Contributions

9.2.1 General Contributions

This thesis has developed a framework for solving multi-robot search problems in the physical world, which includes a number of general contributions:

1. The formalization and hardness analysis of multi-robot search in the physical world for both adversarial and non-adversarial targets.

2. The first integrated framework for solving efficient search, guaranteed search, and constrained search with the following virtues:

    (a) Operates without a centralized controller.

    (b) Scalable to large teams and physical environments.

    (c) Incorporates online information through replanning.

    (d) Capable of operating in partially known environments.

    (e) Operates under the restrictions of limited communication.

    (f) Incorporates subsequent actions into search tasks.

3. The design of a decentralized architecture for testing and analysis of search algorithms. Verification of the proposed framework in both simulation and on heterogeneous search teams consisting of both humans and robots.

4. Application of proposed algorithms to the MAGIC 2010 competition search and secure problem.

9.2.2 Algorithmic and Theoretical Contributions

The specific algorithmic and theoretical contributions are listed below:

The first scalable, online algorithm for solving the multi-robot efficient search problem that achieves performance bounds on the finite-horizon: Finite-Horizon Path Enumeration (FHPE+SA) was shown to perform competitively with general POMDP solvers at a fraction of the computational cost, was validated using data from ranging sensors to a mobile target, and was proven to provide performance guarantees due to the submodularity of the efficient search objective function.

The first anytime algorithm for multi-robot guaranteed search that finds solutions with fewer searchers with increasing runtime: Guaranteed Search with Spanning Trees (GSST) was shown to find solutions with fewer searchers than competing algorithms, and was proven to be probabilistically complete for internal, monotone, and connected search schedules. An extension (G-GSST) allows for the optimization of clearing time, and was shown to outperform several competing guaranteed search methods.

The first algorithm that combines worst-case guarantees for an adversarial target with average-case performance considering a non-adversarial target: The combined efficient/guaranteed search resource allocation algorithm uses FHPE+SA and G-GSST as components and is generalizable to incorporate other methods. The algorithm bridges the gap between risky average-case search and conservative worst-case search, and was demonstrated on a human-robot team performing a search task in an office building.

The introduction of Periodic Connectivity (MIPP-PC) as a method for connectivity maintenance for multi-robot coordination: An extension of receding horizon planning allows the incorporation of periodic connectivity constraints into search tasks. The algorithm was shown to achieve lower expected capture times than techniques that require continual connectivity, and was proven to provide loose performance guarantees, which are limited by the hardness of connectivity constrained planning problems.

The first conservative fusion rule for the efficient search problem: The minimum data fusion rule recovers the underlying distribution without overestimating the probability of capture, was shown to provide performance competitive with all-to-all communication in several scenarios, and was proven to yield a conservative estimate in the efficient search domain.

The introduction of subsequent action costs into the search problem: FHPE+SA was extended to incorporate a subsequent action, and was shown to improve the performance of a state-of-the-art mobile manipulator performing a fetch task. It was also proven that search and subsequent actions can be decoupled for the case of a stationary target with an undiscounted reward metric.

9.2.3 Software Contributions

In addition to the algorithmic and theoretical contributions, a software package "Guaranteed Search with Spanning Trees" was developed and made publicly available. The package contains implementations of spanning tree generation and traversal methods as well as general graph search tools. The software is available at the following url:

`http://www.frc.ri.cmu.edu/projects/emergencyresponse/ghthesis`

9.3 Avenues for Future Work

The results in this thesis open up a number of avenues for improving search tasks and solving additional problems in robotics, computer science, and mathematics.

9.3.1 Environment Decomposition

Discretization of environments for search tasks is an active research area. Automatic discretization using the constrained Delaunay triangulation (Shewchuk, 2002) or the Voronoi diagram (Kolling and Carpin, 2008) are two possible approaches. Visibility-based methods that take advantage of environment geometry are an alternative (Guibas et al., 1999), but a visibility-based method that produces a sufficiently small number of convex cells in complex, cluttered environments is still an open problem. In some cases, a larger number of cells may actually require fewer searchers, which leaves open the question of how best to discretize to help minimize the number of searchers. In addition, alternative methods for taking into account varying sensor modalities (e.g., limited range and limited FOV) yield interesting avenues for future work.

9.3.2 Efficient Search

One extension of FHPE+SA is to apply the algorithm to the case where actions of searchers are no longer fully deterministic. For instance, a searcher may have a fifty percent chance of failing to move because of rubble blocking the way. The POMDP formulation of MESPP can easily express this scenario. The solution to this POMDP would no longer be a deterministic searcher path, but it would instead be a distribution over paths. Even though submodular set analysis does not apply directly, the resulting objective function on distributions over paths may still show qualities related to submodularity, leading to theoretical guarantees for sequential allocation. Singh et al. (2009b) have developed some preliminary results in this direction.

One important avenue for future work is a more comprehensive analysis of the communication requirement of implicitly coordinated solutions. For moderately sized teams, such as those examined in this paper, full broadcast communication is still tractable, but increasing team size can lead to a communication bottleneck. Distributed communication solutions have been proposed in the POMDP literature, which could alleviate this problem for large implicitly coordinated teams (Roth, 2007).

9.3.3 Guaranteed Search

In Chapter 4, GSST selected trees randomly using one of two methods. Alternatively, informed heuristics could be used to generate spanning trees with a high likelihood of yielding clearing schedules with few searchers. GSST could also incorporate traversal strategies other than the ones presented in this thesis. One potential avenue to explore is the use of a traversal based on a graph cut algorithm similar to the one used by Kolling and Carpin (2010) for weighted graph searching on trees.

In addition, GSST does not provide a bound on the current solution quality relative to optimal. In other words, when the search is stopped, we cannot say whether or not we have found a minimal (or near-minimal) search schedule. Solving this problem requires a method for bounding the minimum number of searchers from below. One possible approach is to combine GSST with a SAT solver (see Appendix C) that provides a lower bound. Another possibility is to use a relaxation of an integer programming formulation to generate a lower bound, possibly in conjunction with branch and bound techniques. One of the main challenges in generating near-optimal performance guarantees for GSST is to do so while keeping the algorithm linearly scalable, distributable, modifiable online, and preserving of anytime capabilities.

An important open problem regarding GSST is the completeness of labeled traversal. In other words, can we guarantee that at least one minimal labeled traversal exists after having examined all spanning trees of a graph. It is possible to construct worst-case graphs where a minimal labeled traversal for a particular starting node is not possible (Kehagias et al., 2009a), but such an example has not been found if the starting node is left unspecified.

GSST restricts the moves of the searchers to those that do not recontaminate nodes in the graph. Node recontamination occurs if a searcher leaves a node unguarded, and one or more of its adjacent nodes are dirty (with the exception of the node the searcher is moving into). Even if all starting nodes are considered, recontamination can improve the search number for connected search on graphs (Yang et al., 2004). However, situations in which recontamination helps may be rare in realistic environments, making it difficult to incorporate them into guaranteed search. The approach used by the IGNS algorithm (Kehagias et al., 2009b) would be one place to start.

9.3.4 Combining Efficient and Guaranteed Search

The algorithm presented in Chapter 5 does not directly use a weighting variable α to incorporate confidence in the model. Instead, it caches many solutions and allows the user to choose one at runtime in an anytime fashion. One method for directly incorporating α would be first to determine the lowest number of searchers capable of clearing and assign the remainder of the searchers proportional to α. An alternative would be to have searchers switch between G-GSST and FHPE+SA during the schedule with some probability related to α. This would allow for dynamic switching but would require careful tuning of the switching function.

Chapter 5 assumes that the searchers have a "perfect" sensor that will always detect the target if it resides in the same cell. This assumption can be relaxed somewhat by modeling a non-unity capture probability into the average-case reward function. For the worst-case reward function, the searchers could run the schedule several times to generate a bound on the worst-case probability of missing the target (i.e., each schedule will be guaranteed to have some nonzero probability of locating the target). More sophisticated modeling approaches could also be applied to determine a probabilistic worst-case guarantee with imperfect sensors (Kolling and Carpin, 2009).

The algorithms in Chapter 5 provide strategies robust to a worst-case adversarial target as well as a target with a known motion model. In some cases, however, the target may not be perfectly adversarial. This scenario requires modeling a range of targets from fully non-adversarial to perfectly adversarial. Recent preliminary work by Strom et al. (2010) extends the ideas in this thesis to consider such a range of targets. Further analysis in this direction has the potential to provide a tighter integration of efficient and guaranteed search.

9.3.5 Connectivity and Communication

Extending the techniques in Chapter 6 to partially known and dynamic environments presents a number of challenges. In the case of dynamic environments, it may be necessary to develop contingency plans if the network does not regain connectivity due to changes in the environment. In addition, further analysis of possible performance guarantees in constrained planning domains is a rich area for theoretical study.

The development of a decentralized data fusion rule for a large number of distributions is an interesting area for future work. Problems such as active estimation (e.g., with range-only sensors (Hollinger et al., 2008) or wireless signal strength (Ferris et al., 2006)) would benefit from conservative data fusion rules as well. The minimum fusion rule presented in Chapter 6 is computationally simple and easy to implement for any probabilistic optimization problem. However, even though it is a conservative fusion rule for the efficient search problem, this is not the case when applied to arbitrary probability distributions. Without a trapping capture state, renormalization could cause a non-conservative estimate. Alternative fusion rules are available if the estimate is Gaussian (Makarenko and Durrant-Whyte, 2004), but such an assumption is not valid for many interesting problems.

9.3.6 Combining Search and Subsequent Action

The results in Chapter 7 motivate a more extensive analysis of environments and tasks in which search and action can be decoupled. In the fetch domain, search and subsequent action could be heuristically decoupled even when they could not be provably decoupled. However, this trend may not be the case in other domains.

In addition, the development of a framework for ordering and solving multiple queries is of significant interest. If multiple targets need to be found and multiple searchers are available, a searcher could potentially complete the action task and then rejoin the search for the remaining targets. Extending the approach presented in this thesis to cope with such scheduling would enable autonomous systems to provide continuous assistance for daily living.

9.3.7 Target Tracking

This thesis deals primarily with the search problem, where the objective is to find a target. A related problem is target tracking, where contact to a target must be maintained over a period of time. An open problem is the development of a unified search and tracking framework that allows for continuous surveillance of objects of interest. One promising avenue toward this goal is to extend the work of Webb and Furukawa (2006) to multiple searchers. Another approach would be to extend the search and subsequent action framework from Chapter 7 to interleave tracking and search.

9.3.8 Planning under Uncertainty

This thesis showed that efficient search can be expressed as a Partially Observable Markov Decision Process (POMDP), and results demonstrated that FHPE+SA outperforms general POMDP heuristic solutions (see Chapter 3). These results are limited in scope to the efficient search domain, but the general principles behind FHPE+SA are applicable to many POMDP planning problems. Receding horizon planning would likely perform well wherever the underlying cost function is reasonably smooth and has a gradient to follow towards high reward. Sequential allocation is somewhat more difficult to apply to POMDP planning domains, due to the semantics of "sharing" plans. In many domains, it would be more appropriate to share the distributions that result from executing a plan. A similar coordination idea was developed in the data fusion community for distributed path planning (Grocholsky, 2002). Developing optimal or near-optimal methods for generating predicted distributions from executed team paths in POMDP domains is an interesting avenue for future research.

9.3.9 Implicit Coordination

This thesis focuses directly on the application of implicit coordination methods to multi-robot search problems. However, the principles behind implicit coordination are applicable to a number of problems other than multi-robot search, including coverage, exploration, tracking, monitoring, and information gathering. These related multi-robot domains demonstrate the broader impact of the work in this thesis.

The problem of completely covering an environment with one or more sensors in the minimum time is closely related to search. Coverage differs from efficient search in that the entire area must be observed, and it differs from guaranteed search in that a frontier does not need to be maintained. Recent work in the coverage domain has led to scalable multi-robot coordination algorithms (Zheng et al., 2005; Agmon et al., 2008), some of which allow for decentralized operation and have performance guarantees. The insights gained in Chapter 3 regarding the relationship between diminishing returns and performance guarantees may be applicable to further the analysis of multi-robot coverage algorithms. In addition, many coverage algorithms utilize an underlying spanning tree (or trees) to guide the coverage task. The spanning tree generation techniques developed for GSST in Chapter 4 could improve the generation of spanning trees for coverage tasks as well.

Another closely related problem is the active estimation of a target using a mobile multi-sensor system. For instance, a number of robots may be equipped with range and/or bearing sensors, and they must minimize the tracking error of a target whose movement they cannot control. In some cases, the robots will also need to estimate their own positions during tracking through the use of inter-robot measurements (Tully et al., 2009). The challenge lies in configuring the team to localize itself and to ensure that the estimate of the target's position remains accurate in subsequent time steps. This problem becomes even more difficult in cluttered environments with obstacles, where the teams' movements are heavily restricted. Both implicit coordination and receding-horizon planning techniques are applicable to this scenario; however, the measurements between the robots break the diminishing returns property. Preliminary results have shown that algorithms like FHPE+SA (from Chapter 3) perform reasonably well in these domains (Djugash et al., 2010), but further advances in algorithm design and theoretical analysis could potentially improve performance.

Multi-robot search can be considered a special case of a broader set of information gathering problems (Singh et al., 2009a). The general information gathering framework includes such tasks as monitoring scientific phenomena, mapping unknown environments, and measuring quantities like temperature, air quality, water quality, etc. If the phenomena that need to be measured are changing, an increasingly large number of sensor are required as area size increases. In these cases, mobile sensors can improve performance significantly. Depending on the objective function, the resulting path optimization problem may maintain the quality of diminishing returns, which would allow the straightforward application of implicit coordination. In some domains, additional techniques, such as the use of spanning trees or relaxing of constraints, would be required to increase the coordination between the mobile sensors. In all cases, examining the formal properties of the objective function, as done throughout this thesis, can guide algorithm design for information gathering tasks.

Challenges arise in the domains described above, since determining a good representation can be problem specific and often nontrivial. In addition, the need for dealing with heterogeneity in team capabilities and complex interactions between agents can reduce the effectiveness of implicitly coordinated solutions for many multi-robot problems. However, this thesis has shown that several search problems that at first seemed outside the reach of implicit coordination can be solved efficiently by relaxing constraints or improving the environment representation. The theoretical and algorithmic contributions in this thesis have given researchers a strong foundation to continue analysis of multi-robot coordination problems and techniques. Leveraging these results, we can achieve scalable, decentralized, and robust multi-robot systems capable of operating in the ever-changing physical world.

Bibliography

Adler, M., Racke, H., Sivadasan, N., and Sohler, C. (2003). Randomized pursuit-evasion in graphs. *Combinatorics, Probability, and Computing*, 12(3):225–244.

Agmon, N., Hazon, N., and Kaminka, G. (2008). The giving tree: Constructing trees for efficient offline and online multi-robot coverage. *Annals of Mathematics and AI*, 52(2–4):143–168.

Alspach, B. (2006). Searching and sweeping graphs: A brief survey. *Matematiche*, 59:5–37.

Anisi, D. (2009). *On Cooperative Surveillance, Online Trajectory Planning and Observer Based Control*. PhD thesis, Royal Institute of Technology (KTH), Stockholm, Sweden.

Anisi, D., Ögren, P., and Hu, X. (2008). Cooperative surveillance missions with multiple UGVs. In *Proc. IEEE Conf. Decision and Control*.

Barrière, L., Flocchini, P., Fraigniaud, P., and Santoro, N. (2002). Capture of an intruder by mobile agents. In *Proc. ACM Symp. Parallel Algorithms and Architectures*.

Barrière, L., Fraigniaud, P., Santoro, N., and Thilikos, D. (2003). Searching is not jumping. *Graph-Theoretic Concepts in Computer Science*, 2880:34–45.

Bienstock, D. and Seymour, P. (1991). Monotonicity in graph searching. *J. Algorithms*, 12(2):239–245.

Borie, R., Tovey, C., and Koenig, S. (2009). Algorithms and complexity results for pursuit-evasion problems. In *Proc. Int. Joint Conf. Artificial Intelligence*.

Bourgault, F., Furukawa, T., and Durrant-Whyte, H. (2003). Coordinated decentralized search for a lost target in a bayesian world. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*.

Bourgault, F., Furukawa, T., and Durrant-Whyte, H. (2006). Optimal search for a lost target in a bayesian world. *Field and Service Robotics*, 24:209–222.

Calisi, D., Farinelli, A., Locchi, L., and Nardi, D. (2007). Multi-objective exploration and search for autonomous rescue robots. *J. Field Robotics*, 24(8–9):763–777.

Calliess, J.-P. and Gordon, G. (2008). No-regret learning and a mechanism for distributed multi-agent planning. In *Int. Conf. Autonomous Agents and Multi-Agent Systems*.

Carlin, A. and Zilberstein, S. (2009). Myopic and non-myopic communication under partial observability. In *Proc. IEEE Conf. Intelligent Agent Technology*.

Char, J. (1968). Generation of trees, two-trees, and storage of master forests. *IEEE Trans. Circuit Theory*, 15(3):228–238.

Defence, Science and Technology Organisation, Land Operations Division (2009). Multi Autonomous Ground-robotic International Challenge: Guidelines. `http://www.dsto.defence.gov.au/MAGIC2010/`.

Dendris, N., Kirousis, L., and Thilikos, D. (1994). Fugitive-search games on graphs and related parameters. In *Proc. Int. Workshop Graph-Theoretic Concepts in Computer Science*.

Diankov, R. and Kuffner, J. (2008). OpenRAVE: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Carnegie Mellon Univ.

Djugash, J., Hollinger, G., and Singh, S. (2010). Reducing drift in cooperative target tracking. Working Paper.

Djugash, J., Singh, S., and Grocholsky, B. (2008). Decentralized mapping of robot-aided sensor networks. In *Proc. IEEE Int. Conf. Robotics and Automation*.

Djugash, J., Singh, S., Kantor, G., and Zhang, W. (2006). Range-only SLAM for robots operating cooperatively with sensor networks. In *Proc. IEEE Int. Conf. Robotics and Automation*.

Eaton, J. and Zadeh, L. (1962). Optimal pursuit strategies in discrete-state probabilistic systems. *J. Basic Engineering*, 84:23–28.

Emery-Montemerlo, R. (2005). *Game-Theoretic Control for Robot Teams*. PhD thesis, Robotics Institute, Carnegie Mellon Univ.

Ferris, B., Fox, D., and Lawrence, N. (2007). WiFi-SLAM using gaussian process latent variable models. In *Proc. Int. Joint Conf. Artificial Intelligence*.

Ferris, B., Hahnel, D., and Fox, D. (2006). Gaussian processes for signal strength-based location estimation. In *Proc. Robotics: Science and Systems Conf.*

Flocchini, P., Huang, M., and Luccio, F. (2007). Decontamination of chordal rings and tori using mobile agents. *Int. J. Foundations of Computer Science*, 18(3):547–564.

Flocchini, P., Huang, M., and Luccio, F. (2008). Decontamination of hypercubes by mobile agents. *Networks*, 52(3):167–178.

Flocchini, P., Nayak, A., and Schulz, A. (2005). Cleaning an arbitrary regular network with mobile agents. In *Proc. Int. Conf. Distributed Computing and Internet Technology*.

Fomin, F., Fraigniaud, P., and Thilikos, D. (2004). The price of connectedness in expansions. Technical Report LSI-04-28-R, UPC Barcelona.

Fomin, F. and Thilikos, D. (2008). An annotated bibliography on guaranteed graph searching. *Theoretical Computer Science*, 399:236–245.

Fraigniaud, P. and Nisse, N. (2006). Connected treewidth and connected graph searching. In *Proc. Latin American Symp. Theoretical Informatics*.

Furukawa, T., Durrant-Whyte, H., and Lavis, B. (2007). The element-based method - theory and its application to bayesian search and tracking. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*.

Gerkey, B. (2004). Research website: Pursuit-evasion with teams of robots. `http://ai.stanford.edu/~gerkey/research/pe/index.html`.

Gerkey, B. and Mataric, M. (2002). Sold!: Auction methods for multi-robot coordination. *IEEE Trans. Robotics and Automation*, 18(5):758–768.

Gerkey, B., Thrun, S., and Gordon, G. (2005). Parallel stochastic hill-climbing with small teams. In *Proc. Int. NRL Workshop Multi-Robot Systems*.

Gerkey, B., Thrun, S., and Gordon, G. (2006). Visibility-based pursuit-evasion with limited field of view. *Int. J. Robotics Research*, 25(4):299–315.

Gerkey, B., Vaughan, R., and Howard, A. (2003). The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proc. Int. Conf. Advanced Robotics*.

Grocholsky, B. (2002). *Information-Theoretic Control of Multiple Sensor Platforms*. PhD thesis, Univ. of Sydney.

Guestrin, C., Krause, A., and Singh, A. (2005). Near-optimal sensor placements in gaussian processes. In *Proc. Int. Conf. Machine Learning*.

Guibas, L., Latombe, J., LaValle, S., Lin, D., and Motwani, R. (1999). Visibility-based pursuit-evasion in a polygonal environment. *Int. J. Comp. Geometry and Applications*, 9(5):471–494.

Hegazy, T. (2004). *A Distributed Approach to Dynamic Autonomous Agent Placement for Tracking Moving Targets with Application to Monitoring Urban Environments*. PhD thesis, Georgia Institute of Technology.

Hespanha, J. and Prandini, M. (2002). Optimal pursuit under partial information. In *Proc. Mediterranean Conf. Control and Automation*.

Hespanha, J., Prandini, M., and Sastry, S. (2000). Probabilistic pursuit-evasion games: A one-step nash approach. In *Proc. IEEE Conf. Decision and Control*.

Hollinger, G., Djugash, J., and Singh, S. (2008). Tracking a moving target in cluttered environments with ranging radios. In *Proc. IEEE Int. Conf. Robotics and Automation*.

Hollinger, G., Ferguson, D., Srinivasa, S., and Singh, S. (2009a). Combining search and action for mobile robots. In *Proc. IEEE Int. Conf. Robotics and Automation*.

Hollinger, G., Kehagias, A., and Singh, S. (2010a). GSST: Anytime guaranteed search. *Autonomous Robots*, 29(1):99–118.

Hollinger, G., Kehagias, A., and Singh, S. (2010b). Improving the efficiency of clearing with multi-agent teams. *Int. J. Robotics Research*, 29(8):1088–1105.

Hollinger, G. and Singh, S. (2010). Multi-robot coordination with periodic connectivity. In *Proc. IEEE Int. Conf. Robotics and Automation*.

Hollinger, G., Singh, S., Djugash, J., and Kehagias, A. (2009b). Efficient multi-robot search for a moving target. *Int. J. Robotics Research*, 28(2):201–219.

Howard, A., Parker, L., and Sukatme, G. (2005). Experiments with a large heterogeneous mobile robot team. *Int. J. Robotics Research.*, 25(5–6):431–447.

Howard, A. and Roy, N. (2003). The robotics data set repository (Radish). `http://radish.sourceforge.net/`.

Hsieh, M. A., Cowley, A., Kumar, V., and Taylor, C. (2008). Maintaining network connectivity and performance in robot teams. *J. Field Robotics*, 26(1–2):111–131.

Hu, L. and Evans, D. (2004). Localization for mobile sensor networks. In *Proc. Int. Conf. Mobile Computing and Networking*.

Isler, V., Kannan, S., and Khanna, S. (2005). Randomized pursuit-evasion in a polygonal environment. *IEEE Trans. Robotics*, 21(5):875–884.

Jain, A. and Kempe, C. (2010). El-E: An assitive robot that fetches objects from flat surfaces. *Autonomous Robots*, 28(1):45–64.

Julier, S. and Uhlmann, J. (2001). General decentralised data fusion with covariance intersection (CI). In Hall, D. and Llinas, J., editors, *Handbook of Data Fusion*, pages 319–343. CRC Press.

Kaelbling, L., Littman, M., and Cassandra, A. (1998). Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134.

Kalra, N. (2006). *A Market-Based Framework for Tightly-Coupled Planned Coordination in Multirobot Teams*. PhD thesis, Robotics Institute, Carnegie Mellon Univ.

Kehagias, A., Hollinger, G., and Gelastopoulos, A. (2009a). Searching the nodes of a graph: Theory and algorithms. Technical Report 0905.3359v1 [cs.DM], arXiv Repository.

Kehagias, A., Hollinger, G., and Singh, S. (2009b). A graph search algorithm for indoor pursuit / evasion. *Mathematical and Computer Modelling*, 50(9–10):1305–1317.

Kloks, T. (1994). *Treewidth: Computations and Approximations*. Springer, Berlin, Germany.

Kolling, A. and Carpin, S. (2008). Extracting surveillance graphs from robot maps. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*.

Kolling, A. and Carpin, S. (2009). Probabilistic graph-clear. In *Proc. IEEE Int. Conf. Robotics and Automation*.

Kolling, A. and Carpin, S. (2010). Pursuit-evasion on trees by robot teams. *IEEE Trans. Robotics*, 26:32–47.

Krause, A. and Guestrin, C. (2007). Near-optimal observation selection using submodular functions. In *Proc. Conf. Artificial Intelligence*.

Krause, A., Guestrin, C., Gupta, A., and Kleinberg, J. (2006). Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proc. Information Processing in Sensor Networks*.

Krause, A., McMahan, B., Guestrin, C., and Gupta, A. (2007). Selecting observations against adversarial objectives. In *Proc. Neural Information Processing Systems*.

Krause, A., McMahan, B., Guestrin, C., and Gupta, A. (2008). Robust submodular observation selection. *J. Machine Learning Research*, 9:2761–2801.

Kumar, V., Rus, D., and Singh, S. (2004). Robot and sensor networks for first responders. *Pervasive Computing*, 3(4):24–33.

Kurniawati, H., Hsu, D., and Lee, W. (2008). SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems Conf.*

LaPaugh, A. (1993). Recontamination does not help to search a graph. *J. ACM*, 40(2):224–245.

Lau, H., Huang, S., and Dissanayake, G. (2005). Optimal search for multiple targets in a built environment. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*.

Lau, H., Huang, S., and Dissanayake, G. (2006). Probabilistic search for a moving target in an indoor environment. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*.

LaValle, S., Lin, D., Guibas, L., Latombe, J., and Motwani, R. (1997). Finding an unpredictable target in a workspace with obstacles. In *Proc. IEEE Int. Conf. Robotics and Automation*.

Lawrence, N. (2005). Probabilistic non-linear principal component analysis with gaussian process latent variable models. *J. Machine Learning Research*, 6:1783–1816.

Lawrence, N. and Quiñonero-Candela, J. (2006). Local distance preservation in the GP-LVM through back constraints. In *Proc. Int. Conf. Machine Learning*.

Liao, E., Hollinger, G., Djugash, J., and Singh, S. (2006). Preliminary results in tracking mobile targets using range sensors from multiple robots. In *Proc. Int. Symp. Distributed Autonomous Robotic Systems*.

Likhachev, M. and Ferguson, D. (2009). Planning long dynamically feasible maneuvers for autonomous vehicles. *Int. J. Robotics Research*, 28(8):933–945.

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., and Thrun, S. (2005). Anytime dynamic A*: An anytime, replanning algorithm. In *Proc. Int. Conf. Automated Planning and Scheduling*.

Littman, M., Cassandra, A., and Kaelbling, L. (1995). Learning policies for partially observable environments: Scaling up. In *Proc. Int. Conf. Machine Learning*.

Makarenko, A. and Durrant-Whyte, H. (2004). Decentralized data fusion and control in active sensor networks. In *Proc. Int. Conf. Information Fusion*.

Megiddo, N., Hakimi, S., Garey, M., Johnson, D., and Papadimitriou, C. (1988). The complexity of searching a graph. *J. ACM*, 35(1):18–44.

Michael, N., Zavlanos, M. M., Kumar, V., and Pappas, G. J. (2008). Maintaining connectivity in mobile robot networks. In *Proc. Int. Symp. Experimental Robotics*.

Multispectral Solutions, Inc. (2008). Company website. `http://www.multispectral.com/`.

Nair, R., Varakantham, P., Tambe, M., and Yokoo, M. (2005). Networked distributed POMDPs: A synthesis of distributed constraint optimization and POMDPs. In *Proc. AAAI Conf. Artificial Intelligence*.

Nourbakhsh, I., Powers, R., and Birchfield, S. (1995). Dervish: An office-navigating robot. *AI Mag.*, 16(2):53–60.

Ong, S., Png, S., Hsu, D., and Lee, W. (2009). POMDPs for robotic tasks with mixed observability. In *Proc. Robotics: Science and Systems Conf.*

Parsons, T. (1976). Pursuit-evasion in a graph. In Alavi, Y. and Lick, D., editors, *Theory and Applications of Graphs*, pages 426–441. Springer, Berlin/Heidelberg.

Pineau, J., Gordon, G., and Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. In *Proc. Int. Joint Conf. Artificial Intelligence*.

Priyantha, N., Balakrishnan, H., Demaine, E., and Teller, S. (2005). Mobile-assisted localization in wireless sensor networks. In *Proc. IEEE INFOCOM*.

Quigley, M., Berger, E., and Ng, A. (2007). STAIR: Hardware and software architecture. In *Proc. AAAI Robotics Workshop*.

Roth, M. (2007). *Execution-Time Communication Decisions for Coordination of Multi-Agent Teams*. PhD thesis, Robotics Institute, Carnegie Mellon Univ.

Roy, N., Burgard, W., Fox, D., and Thrun, S. (1998). Coastal navigation: Robot motion with uncertainty. In *Proc. AAAI Fall Symposium: Planning with POMDPs*.

Roy, N., Gordon, G., and Thrun, S. (2003). Planning under uncertainty for reliable health care robotics. In *Proc. Int. Conf. Field and Service Robotics*.

Roy, N., Gordon, G., and Thrun, S. (2005). Finding approximate POMDP solutions through belief compression. *J. Artificial Intelligence Research*, 23:1–40.

Sarmiento, A., Murrieta-Cid, R., and Hutchinson, S. (2004). A multi-robot strategy for rapidly searching a polygonal environment. In *Proc. Ibero-American Conf. Artificial Intelligence*.

Schwaighofer, A., Grigoras, M., Tresp, V., and Hoffmann, C. (2004). GPPS: A gaussian process positioning system for cellular networks. In *Proc. Conf. Neural Information Processing Systems*.

Shewchuk, J. (2002). Delaunay refinement algorithms for triangular mesh generation. *Computational Geometry: Theory and Applications*, 22(1–3):21–74.

Sim, R. and Roy, N. (2005). Global A-optimal robot exploration in SLAM. In *Proc. IEEE Int. Conf. Robotics and Automation*.

Simmons, R. and Koenig, S. (1995). Probabilistic robot navigation in partially observable environments. In *Proc. Int. Joint Conf. Artificial Intelligence*.

Singh, A., Krause, A., Guestrin, C., and Kaiser, W. (2009a). Efficient informative sensing using multiple robots. *J. Artificial Intelligence Research*, 34:707–755.

Singh, A., Krause, A., Guestrin, C., Kaiser, W., and Batalin, M. (2007). Efficient planning of informative paths for multiple robots. In *Proc. Int. Joint Conf. Artificial Intelligence*.

Singh, A., Krause, A., and Kaiser, W. (2009b). Nonmyopic adaptive informative path planning for multiple robots. In *Proc. Int. Joint Conf. Artificial Intelligence*.

Smith, T. (2007). *Probabilistic Planning for Robotic Exploration*. PhD thesis, Robotics Institute, Carnegie Mellon Univ.

Spaan, M. and Vlassis, N. (2004). Perseus: Randomized point-based value iteration for POMDPs. Technical Report IAS-UVA-04-02, Informatics Institute, University of Amsterdam.

Srinivasa, S., Ferguson, D., Helfrich, C., Berenson, D., Collet, A., Diankov, R., Gallagher, G., Hollinger, G., Kuffner, J., and Weghe, J. V. (2010). HERB: A home exploring robotic butler. *Autonomous Robots*, 28(1):5–20.

Strom, J., Morton, R., Reilly, K., and Olson, E. (2010). Online probabilistic pursuit of adversarial evaders. In *Proc. ICRA Workshop on Search and Pursuit/Evasion in the Physical World*.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press, Cambridge, MA.

Tisdale, J., Kim, Z., and Hedrick, J. (2009). Autonomous path planning and estimation using UAVs. *IEEE Robotics and Automation Mag.*, June.

Tully, S., Kantor, G., and Choset, H. (2009). Leap-frog path design for multi-robot cooperative localization. In *Proc. Int. Conf. Field and Service Robotics*.

Vidal, R., Shakernia, O., Kim, H., Shim, D., and Sastry, S. (2002). Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation. *IEEE Trans. Robotics and Automation*, 18(5):662–669.

Wang, J., Fleet, D., and Hertzmann, A. (2007). Gaussian process dynamical models for human motion. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(2):283–298.

Webb, S. and Furukawa, T. (2006). Belief driven manipulator control for integrated searching and tracking. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*.

Wilson, D. (1996). Generating random spanning trees more quickly than the cover time. In *Proc. ACM Symp. Theory of Computing*.

Yang, B., Dyer, D., and Alspach, B. (2004). Sweeping graphs with large clique number. In *Proc. Int. Symp. Algorithms and Computation*.

Yang, P., Freeman, R., Gordon, G., Lynch, K., Srinivasa, S., and Sukthankar, R. (2008). Decentralized estimation and control of graph connectivity in mobile sensor networks. In *Proc. American Control Conference*.

Zavlanos, M. and Pappas, G. (2008). Distributed connectivity control of mobile networks. *IEEE Trans. Robotics*, 24(6):1416–1428.

Zheng, X., Jain, S., Koenig, S., and Kempe, D. (2005). Multi-robot forest coverage. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*.

Zilberstein, S. (1996). Using anytime algorithms in intelligent systems. *Artificial Intelligence Mag.*, 17(3):73–86.

Zlot, R. and Stentz, A. (2006). Market-based multirobot coordination for complex tasks. *Int. J. Robotics Research*, 25(1):73–101.

Appendix A

Multimedia Extensions

A number of multimedia extensions to this thesis are available at the following url:
`http://www.frc.ri.cmu.edu/projects/emergencyresponse/ghthesis`

1. (Video): Simulated efficient search trials using FHPE+SA on the house and NSH maps (Chapter 3).
2. (Video): Efficient search trials with ranging radio data in Newell-Simon Hall at Carnegie Mellon University (Chapter 3).
3. (Video): Simulated GSST experiments showing clearing schedules on the Gates, NSH, cave, and museum maps (Chapter 4).
4. (Video): Simulated experiments using the combined efficient/guaranteed search algorithm on the NSH map (Chapter 5).
5. (Video): Human-Robot team experiments in Newell-Simon Hall at Carnegie Mellon University demonstrating the combined algorithm (Chapter 5).
6. (Video): Search and coverage experiments with periodic connectivity, both in simulation and on a robot team (Chapter 6).
7. (Video): A mobile manipulator searching for a coffee mug, finding it, picking it up, and placing it in the sink. Filmed at Intel Research Pittsburgh (Chapter 7).
8. (Video): MAGIC 2010 simulation playback showing a successful clearing schedule without any failures (Chapter 8).
9. (Video): MAGIC 2010 simulation playback showing a clearing schedule with a recoverable failure (Chapter 8).
10. (Video): MAGIC 2010 simulation playback showing a failure that resulted in abandoning the clearing schedule and resorting to efficient search to complete the mission (Chapter 8).

Appendix B

Proof of Theorems

B.1 Submodularity of Efficient Search Objective

This section proves Theorem 3.1 from Chapter 3.

Theorem B.1 *Equation B.1 is a nondecreasing, submodular set function. This is the objective function optimized by the MESPP.*

$$F(A) = \sum_{Y \in \Psi} \Pr(Y) F_Y(A), \tag{B.1}$$

where $A \subset N'$ *is a feasible searcher path in* G'*,* Ψ *is the space of all possible target paths,* $\Pr(Y)$ *is the probability of the target taking path* Y*, and* $F_Y(A)$ *is the discounted reward received by path* A *if the target chooses path* Y*.*

Proof Consider a time-augmented version of graph G, $G' = (N', E')$. Level 1 of G' represents the vertices of G at time 1, and directed edges from level 1 connect to reachable vertices at time 2. Extend this graph down to a max time τ. The graph now contains vertices $N' = N \times T$ where $T = \{1, \ldots, \tau\}$. This result extends to the infinite case by making τ arbitrarily large. $\mathbf{P}(N')$ denotes the powerset of N', i.e. the set of (time stamped) node subsets. A function $F : \mathbf{P}(N') \to \Re_0^+$ is called *increasing* iff

$$A \subseteq B \Rightarrow F(A) \leq F(B).$$

It is called *submodular* iff

$$A \subseteq B \Rightarrow F(A \cup C) - F(A) \geq F(B \cup C) - F(B).$$

In the above $A, B \in \mathbf{P}(N')$ and $C = \{(m,t)\} \subseteq N'$ – i.e. C is a singleton. In the following, for a given $Y \subseteq N'$ and any $A \subseteq N'$, we define $t_A = \min\{t : (m,t) \in A \cap Y\}$, $F_Y(A) = \gamma^{t_A}$, with the understanding that $\gamma \in (0,1)$, $\min \emptyset = \infty$, and $\gamma^\infty = 0$.

We now show that for every $Y \subseteq N'$ the function $F_Y(A)$ is nondecreasing and submodular. Take an arbitrary Y and fix it for the proof. Take any $A, B \subseteq N'$ and

any $C = \{(m_0, t_0)\} \subseteq N'$. We have:
$t_A = \min\{t : (m,t) \in A \cap Y\}$,
$t_B = \min\{t : (m,t) \in B \cap Y\}$,
$t_C = \min\{t : (m,t) \in C \cap Y\}$.
$A \subseteq B \Rightarrow \{t : (m,t) \in A \cap Y\} \subseteq \{t : (m,t) \in B \cap Y\} \Rightarrow t_A \geq t_B \Rightarrow$
$F_Y(A) = \gamma^{t_A} \leq \gamma^{t_B} = F_Y(B)$.

Hence $F_Y(\cdot)$ is nondecreasing.

Now, regarding submodularity, note that, since C is a singleton, we have two cases: either $C \cap Y \neq \emptyset$ and so $t_c = t_0 < \infty$; or $C \cap Y = \emptyset$ and so $t_c = \infty$. We examine the two cases separately.

Case I, $t_C < \infty$. In this cases we have three subcases.

1. $t_B \leq t_A \leq t_C$. Then $F_Y(B \cup C) = \gamma^{t_B}$, $F_Y(B) = \gamma^{t_B}$, $F_Y(A \cup C) = \gamma^{t_A}$, $F_Y(A) = \gamma^{t_A}$ and $F_Y(A \cup C) - F_Y(A) = \gamma^{t_A} - \gamma^{t_A} = 0 = \gamma^{t_B} - \gamma^{t_B} = F_Y(B \cup C) - F_Y(B)$.

2. $t_B \leq t_C \leq t_A$. Then $F_Y(B \cup C) = \gamma^{t_B}$, $F_Y(B) = \gamma^{t_B}$, $F_Y(A \cup C) = \gamma^{t_C}$, $F_Y(A) = \gamma^{t_A}$ and $F_Y(A \cup C) - F_Y(A) = \gamma^{t_C} - \gamma^{t_A} > 0 = \gamma^{t_B} - \gamma^{t_B} = F_Y(B \cup C) - F_Y(B)$.

3. $t_C \leq t_B \leq t_A$. Then $F_Y(B \cup C) = \gamma^{t_C}$, $F_Y(B) = \gamma^{t_B}$, $F_Y(A \cup C) = \gamma^{t_C}$, $F_Y(A) = \gamma^{t_A}$ and $F_Y(A \cup C) - F_Y(A) = \gamma^{t_C} - \gamma^{t_A} \geq \gamma^{t_C} - \gamma^{t_B} = F_Y(B \cup C) - F_Y(B)$, since $t_A \geq t_B \Rightarrow \gamma^{t_A} \leq \gamma^{t_B} \Rightarrow -\gamma^{t_A} \geq -\gamma^{t_B}$.

Case II, $t_C = \infty$. Then we have a single subcase: $t_B \leq t_A \leq t_C$ from which follows $F_Y(A \cup C) - F_Y(A) = 0 = F_Y(B \cup C) - F_Y(B)$ as already seen.

In every case the submodularity inequality holds.

The one searcher reward function $F(A_1)$ (where A_1 is the searcher's path) is defined by

$$F(A_1) = \sum_{Y \in \Psi} \Pr(Y) F_Y(A_1),$$

where the summation is over all possible target paths Ψ, and $\Pr(Y)$ is the probability of the path Y. The K-searcher reward function $F(A_1 \cup ... \cup A_K)$ (where A_k is the path of the k-th searcher) is defined by

$$F(A_1 \cup ... \cup A_K) = \sum_{Y \in \Psi} \Pr(Y) F_Y(A_1 \cup ... \cup A_K).$$

We now show that for any $K = 1, 2, ...$ and every $Y \subseteq N'$ the function $F(A_1 \cup ... \cup A_K)$ is nondecreasing and submodular. Nondecreasing submodularity is closed under nonnegative linear combinations (and hence expectations). Here $F(A_1 \cup ... \cup A_K)$ is the expected value of $F_Y(A)$ where $A = A_1 \cup ... \cup A_K$, $F_Y(\cdot)$ is a nondecreasing, submodular function, and the expectation is taken over all possible target paths. ∎

B.2 Completeness of GSST

This section proves Theorem 4.1 from Chapter 4.

Definition Given a graph G and an IMC node clearing schedule $\mathbf{S}$ defined over time $t = 1, 2, \ldots, t_f$. Let $sn(\mathbf{S}, t)$ be the number of searchers required in the schedule at time step t. Let $\overline{sn}(\mathbf{S})$ be the maximum number of searchers required for the schedule.[1]

Definition Given a graph G and a search $\mathbf{S}$ of G, the *frontier* at t (under $\mathbf{S}$) is

$$N_F(t) = \{u : u \in N_C(t) \text{ and } \exists v : v \in N_D(t), uv \in E\},$$

i.e., the clear nodes which are connected to dirty nodes.

Lemma B.1 *Given a graph $G = (N, E)$ and a rooted IMC node clearing search* $\mathbf{S}$ *of G. The clearing moves of* $\mathbf{S}$ *generate a sequence of trees,* $(\mathbf{T}_0, \mathbf{T}_1, \mathbf{T}_2, ..., \mathbf{T}_{|N|})$, *where (for $n = 1, 2, ..., |N|$)* $\mathbf{T}_n = (N_n, E_n)$ *and the following hold:*

- ***D1*** $\mathbf{T}_0$ *is the empty graph (and $N_0 = \emptyset$, $E_0 = \emptyset$).*
- ***D2*** *$N_{|N|} = N$, $E_{|N|} \subseteq E$;*
- ***D3*** *for $n = 1, 2, ..., |N|$: $N_{n-1} \subseteq N_n \subseteq N$, $E_{n-1} \subseteq E_n \subseteq E$ (in other words* $\mathbf{T}_{n-1}$ *is a subtree of* $\mathbf{T}_n$*);*
- ***D4*** *for $n = 1, 2, ..., |N|$: $N_n = N_{n-1} \cup \{u_n\}$, and for $n = 2, 3, ...$: $E_n = E_{n-1} \cup \{u_i u_n\}$, with $i \in [1, n-1]$.*

Proof Inductively. Since $\mathbf{S}$ is monotone, it involves $|N|$ clearing moves. $\mathbf{T}_0$ is the empty graph. $\mathbf{T}_1$ is formed by the first move of $\mathbf{S}$, which consists in placing a searcher at the root node. So $\mathbf{T}_1$ is the tree with a single node and trivially has $\mathbf{T}_0$ as a subgraph. Suppose **D3** and **D4** hold up to $m = n$ and consider the $(n+1)$-th clearing move of $\mathbf{S}$. Since $\mathbf{S}$ is connected, we add to $\mathbf{T}_n$ one node u_{n+1} and *exactly* one edge $u_i u_{n+1}$ (with $i \in [1, n]$) to obtain a new tree $\mathbf{T}_{n+1}$ (of $n+1$ nodes and n edges), which also satisfies **D3** and **D4**. The edge $u_i u_{n+1}$ cannot cause a cycle since it was not in the cleared set, and thus not in the tree. Hence **D3** and **D4** hold for $m = 1, 2, ..., |N|$. At $m = |N|$, $N_{|N|}$ contains $|N|$ nodes, hence $N_{|N|} = N$; since $\mathbf{T}_{|N|}$ is a tree, it is a spanning tree of G. ∎

Corollary B.1 *Given a graph $G = (N, E)$, every minimal IMC node clearing search* $\mathbf{S}$ *of G generates a sequence of trees which satisfy the conditions of Lemma B.1.*

[1]Note that $\overline{sn}(\mathbf{S}) = sn(\mathbf{S}, t_f)$ in an IMC clearing schedule since no searchers are removed.

Lemma B.2 *Given a graph $G = (N, E)$ and a tree sequence $\left(\mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_{|N|}\right)$. Then GSST-R / GSST-LW using a single uniformly generated spanning tree has a nonzero probability of producing a search $\mathbf{S}$ which generates $\left(\mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_{|N|}\right)$.*

Proof The probability of generating the tree sequence $\left(\mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_{|N|}\right)$ is:

$$\Pr\left(\mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_{|N|}\right) = \left[\prod_{n=1}^{|N|} \Pr\left(\mathbf{T}_n | \mathbf{T}_{|N|}, \mathbf{T}_0, ..., \mathbf{T}_{n-1}\right)\right] \Pr\left(\mathbf{T}_0 | \mathbf{T}_{|N|}\right) \Pr\left(\mathbf{T}_{|N|}\right).$$

The main idea is that given a tree, there is a nonzero chance of exploring it in any feasible order. Note that the conditioning in the above expression *always* includes $\mathbf{T}_{|N|}$, since this is the first choice made in running GSST-R or GSST-LW. Now obviously, $\Pr\left(\mathbf{T}_0 | \mathbf{T}_{|N|}\right) = 1$. $\Pr\left(\mathbf{T}_{|N|}\right) > 0$ is nonzero for any uniform spanning tree generator.
$\Pr\left(\mathbf{T}_n | \mathbf{T}_{|N|}, \mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_{n-1}\right)$ is the probability of expanding (at the n-th step) $\mathbf{T}_{n-1}$ by the edge $u_i u_n \in \mathbf{E}_n - \mathbf{E}_{n-1}$ which, by the construction of both GSST-R and GSST-LW, is always positive. Finally, $\Pr\left(\mathbf{T}_{|N|} | \mathbf{T}_{|N|}, \mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_{|N|-1}\right) = 1$. Hence $\Pr\left(\mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_{|N|}\right) > 0$ for every sequence $\mathbf{T}_1, ..., \mathbf{T}_{|N|}$. ∎

Lemma B.3 *Given a graph $G = (N, E)$ and a rooted IMC node clearing search $\mathbf{S}$ of G; let $\left(\mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_{|N|}\right)$ be the tree sequence generated by $\mathbf{S}$. Let $\mathbf{S}'$ be a search produced by either GSST-R or GSST-LW and* also *generating $\left(\mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_{|N|}\right)$. Then $\overline{sn}(\mathbf{S}) \geq \overline{sn}(\mathbf{S}')$.*

Proof The proof is exactly the same for GSST-R and GSST-LW, so we only prove the first one, by induction. Let $t_1, ..., t_{|N|}$ be the clearing times of $\mathbf{S}$ and $t'_1, ..., t'_{|N|}$ be the clearing times of $\mathbf{S}'$. Also let $t_0 = t'_0 = 0$.

At $t_0 = t'_0 = 0$ we have $sn(\mathbf{S}', 0) = sn(\mathbf{S}, 0) = 0$.

The only times at which $sn(\mathbf{S}', t)$ may change are $1, t'_1 + 1, ..., t'_{|N|-1} + 1$. Suppose that

$$sn(\mathbf{S}, t_n) \geq sn(\mathbf{S}', t'_n).$$

Further, suppose that at $t'_n + 1$ a new searcher is introduced in $\mathbf{S}'$. This can only happen (in the $\mathbf{S}'$ search) if all of the following hold:

1. at t'_n exactly $|N_F(t)|$ searchers exist in G;

2. there are no searchers inside nodes $u \in N_C(t'_n) \setminus N_F(t'_n)$ (i.e., all searchers are located inside frontier nodes);

3. all searchers are stuck (i.e., moving a searcher out of a frontier node u exposes u to recontamination).

The sequence $\left(\mathbf{T}_0, \mathbf{T}_1, \ldots, \mathbf{T}_{|N|}\right)$ and the clearing times determine the frontier $N_F(t)$ for every t. Hence $\mathbf{S}'$ at t'_n has the same frontier as $\mathbf{S}$ at t_n. If conditions 1-3 above hold in $\mathbf{S}'$, then every searcher is located in a frontier node and is stuck. It is possible that non-stuck searchers exist in $\mathbf{S}$ (located either in frontier or non-frontier nodes) but this also means that $sn(\mathbf{S}, t_n) \geq sn(\mathbf{S}', t'_n) + 1$; hence adding a searcher in $\mathbf{S}'$ at $t'_n + 1$ preserves

$$sn(\mathbf{S}, t_n) \geq sn(\mathbf{S}', t'_n + 1).$$

Since no searchers are added in $\mathbf{S}'$ for $t \in \left[t'_n + 2, t'_{n+1}\right]$ and no searchers are ever removed in $\mathbf{S}$ (i.e., $sn(\mathbf{S}, t_{n+1}) \geq sn(\mathbf{S}, t_n)$) we also get

$$sn(\mathbf{S}, t_{n+1}) \geq sn\left(\mathbf{S}', t'_{n+1}\right).$$

From the above inequality inductively we get $sn\left(\mathbf{S}, t_{|N|}\right) \geq sn\left(\mathbf{S}', t'_{|N|}\right)$, which proves the Lemma. ∎

Theorem B.2 *At each iteration, both random traversal (GSST-R) and label-weighted random traversal (GSST-LW) of a uniformly generated spanning tree have a nonzero chance of yielding a minimal monotone/connected node search schedule on any graph.*

Proof Restating, we must prove that given a graph $G = (N, E)$:

1. GSST-R will generate a minimal monotone, connected clearing of G with probability greater than or equal to $1 - \alpha_1^M$ where M is the number of iterations and $\alpha_1 \in (0, 1)$.

2. GSST-LW will generate a minimal monotone, connected clearing of G with probability greater than or equal to $1 - \alpha_2^M$ where M is the number of iterations and $\alpha_2 \in (0, 1)$.

The proof is exactly the same for GSST-R and GSST-LW, so we only prove the first one. G has at least one minimal rooted IMC node clearing search $\mathbf{S}$ of G. Let $\left(\mathbf{T}_0, \mathbf{T}_1, ..., \mathbf{T}_{|N|}\right)$ be the tree sequence generated by $\mathbf{S}$. By Lemma B.2, GSST-R has a nonzero probability, call it β_1, of generating *in a single iteration* a search $\mathbf{S}'$ with the same tree sequence as $\mathbf{S}$. Then, by Lemma B.3,

$$\overline{sn}(\mathbf{S}) \geq \overline{sn}(\mathbf{S}').$$

Since $\mathbf{S}$ is minimal, $\overline{sn}(\mathbf{S}) = \overline{sn}(\mathbf{S}')$ and so $\mathbf{S}'$ is minimal too. Now, the probability of *not* generating $\mathbf{S}'$ in a single iteration is $\alpha_1 = 1 - \beta_1$; and the probability of *not* generating $\mathbf{S}'$ in M iterations is $\alpha_1^M = (1 - \beta_1)^M$, while the probability of generating $\mathbf{S}'$ in M iterations is $1 - \alpha_1^M$. ∎

Appendix C

SAT Formulation for Guaranteed Search

In this appendix, a reduction from the guaranteed search problem to satisfiability is given. This method could be used to generate a lower bound on the number of searchers (i.e., when an unsatisfiable result is returned with K searchers, the search number must be greater than K). Results are shown using the MiniSAT solver on several graphs that test effectiveness of this reduction for determining minimal search schedules. The current formulation and solver are unable to generate solutions for many of the large graphs used in this thesis.

C.1 Guaranteed Search SAT Problem Description

C.1.1 Graph Search

This section provides a brief summary of the guaranteed search problem discussed at length in Chapter 4. We are given K searchers and a graph $G = (N, E)$ with $|N|$ nodes and $|E|$ edges.[1] At a given discrete time t, the searchers reside on the nodes of the graph and can move to another node at time $t + 1$. If searchers can only move along the edges, then the search schedule is considered *internal*. An evader resides on the graph. If the evader resides on the nodes of the graph, we refer to the problem as *node search*. If the evader resides on the edges of the graph, we refer to it as *edge search*.

The searchers locate the evader if one of them resides on the same node (or slide along the same edge in edge search). The evader is adversarial, arbitrarily fast, and omniscient of the searchers' locations, movements, and plans. The searchers' goal is to move in such a way that no evader can escape them. This is called *clearing* the

[1]For brevity, we will refer to the sets N, K, T and the cardinality of these sets, $|N|$, $|K|$, and $|T|$ interchangeably where it will not cause confusion.

graph. The *search number* of the graph $s(G)$ is the smallest number of searchers that can clear the graph.[2]

At a given time t, there is a set of nodes $N_D(t) \subseteq N$ in which the evader may still reside (conversely edges $E_D(t) \subseteq E$ in edge search). We will refer to this set as the *dirty set*. Similarly, the *cleared set* $N_C(t) \subseteq N$ is the set of nodes in which the evader cannot reside. A graph is cleared at time T when $N_C(T) = N$. If a node (or edge) never returns to the dirty set after being in the cleared set, the schedule is called *monotonic*. If the cleared set forms a connected subgraph of G at all times, the search is called *connected*.

C.1.2 SAT

A SAT formula is a propositional logic formula containing variables and clauses. A variable x_i can be either true or false. A literal l_i is the true assignment of a variable x_i or its negation $\overline{x}_i$. A clause is a disjunction of literals, and a SAT formula in CNF form is a conjunction of clauses. A literal x_i is satisfied if the corresponding variable is true, and conversely a literal $\overline{x}_i$ is satisfied if the corresponding variable takes on the value of false. If any literal in a clause is satisfied, then the clause is satisfied. A SAT formula is satisfied if all of its clauses are satisfied.

We will now reduce a given node search problem on $G = (N, E)$ with K searchers to a SAT formula in CNF form that is satisfiable if and only if $s(G) \leq K$. In addition, a satisfying assignment must return a valid clearing schedule for the searchers.

C.2 SAT Reduction

Assume that we are given K searchers and a graph $G = (N, E)$. The formulation also requires choosing a maximum time T. For multiple searchers moving in one time step, this value can be set to $(N/K + 1)$ or greater. For a single searcher moving at a time, this must be set to $(N + 1)$ or greater.

C.2.1 Variables

The first set of variables describes the cleared and dirty sets at all times. We define the variables c_{nt} as true iff node n is clear at time t. These are defined over the sets $t \in \{1, \ldots, T\}$ and $n \in \{1, \ldots, N\}$. This generates exactly NT variables.

The second set of variables describes the locations of the searchers at all times. We define the variables x_{nkt} as true iff searcher k resides in node n at time t. These are defined over the sets describe above and $k \in \{1, \ldots, K\}$. This generates exactly NTK variables. Thus, the total number of variables in the reduction is $NT + NTK$,

[2]The term $s(G)$ is used in the general sense. However, the search number may change depending on the type of search performed (e.g., node, edge, internal, connected, monotonic).

where N is the number of nodes in the graph, T is the maximum time, and K is the number of searchers.

C.2.2 Clauses

The first set of clauses ensures that all nodes are dirty at the beginning and that all nodes are clear at the end. This generates $2N$ clauses. These clauses are simply:

$$[\overline{c}_{n1}] \ ; \ \forall n \in N$$

$$[c_{nT}] \ ; \ \forall n \in N$$

The next set of clauses are "at least one" clauses constraining the searchers to be on a node in the graph at all time steps $t > 1$. At $t = 1$ we leave the searchers locations unconstrained to allow for all nodes to start out dirty. The corresponding NTK clauses are:

$$[x_{1kt} \vee x_{2kt} \vee \ldots \vee x_{Nkt}] \ ; \ \forall k \in K, \forall t \in T, t > 1$$

The searchers movements also require N^2TK "at most one" clauses excluding searchers from being on more than one node; these clauses do not, however, restrict the searchers from "teleporting" from one time step to the next:

$$[\overline{x}_{ikt} \vee \overline{x}_{jkt}] \ ; \ \forall k \in K, \forall t \in T, t > 1, \forall i, j \in N, i \neq j$$

The clearing rules are defined by NTK clauses ensuring that all nodes containing a searcher are cleared:

$$[\overline{x}_{nkt} \vee c_{nt}] \ ; \ \forall n \in N, \forall k \in K, \forall t \in T, t > 1$$

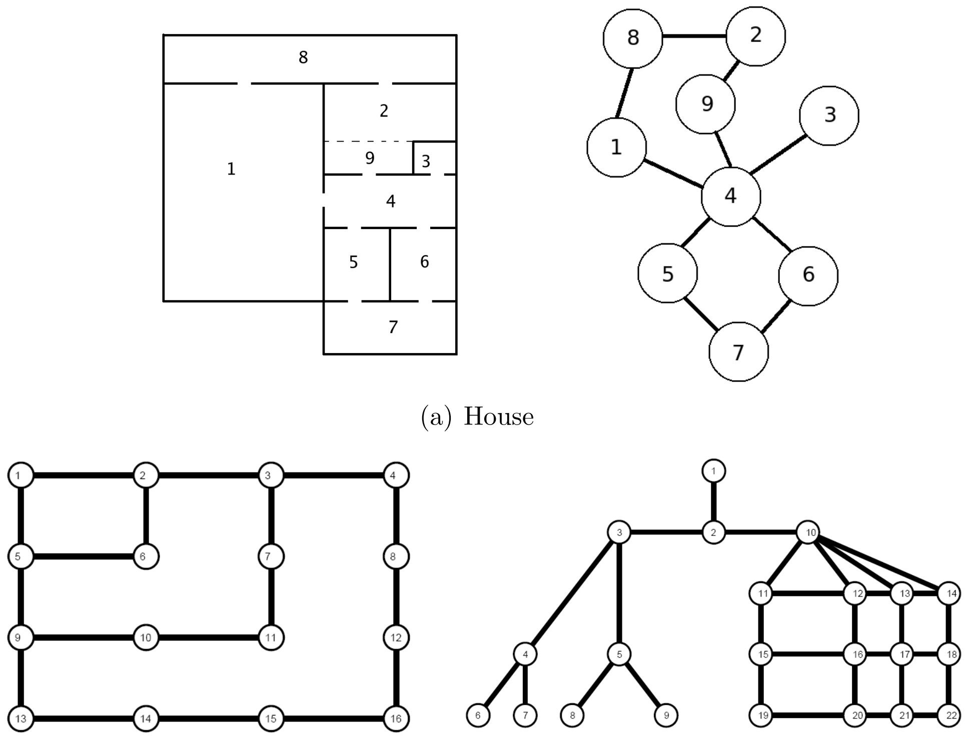
The following NT clauses force nodes that are dirty to stay dirty unless cleared by a searcher:

$$[\overline{c}_{nt} \vee c_{n(t-1)} \vee x_{1nt} \vee \ldots \vee x_{Knt}] \ ; \ \forall n \in N, \forall k \in K, \forall t \in T, t > 1$$

Finally, NTB clauses force unguarded nodes to become dirty if any adjacent nodes are dirty:

$$[\overline{c}_{nt} \vee c_{at} \vee x_{1nt} \vee \ldots \vee x_{Knt}] \ ; \ \forall n, a \in N, na \in E, \forall t \in T, t > 1$$

This formulation generates, in total, $2N + 2NTK + N^2TK + NT + NTB$ clauses, where N is the number of nodes in the graph, T is the maximum time, K is the number of searchers, and B is the average branching factor of the graph. Table C.1 gives a summary of the SAT instance sizes for various graphs. Many of these graphs



(a) House

(b) Loops

(c) Tree/Grid

Figure C.1: Simple graphs used to test the SAT formulation of guaranteed search.

Table C.1: Information for satisfiability on various maps

Map	$s(G)$	Nodes	Edges	Max Time	Variables	Clauses
House	2	9	10	5	135	502
Loops	3	16	18	7	448	2810
Tree/grid	4	22	30	7	770	6632
Gates	3	32	35	100	12800	167,275
Cave	3	43	48	100	17200	295,106
NSH	3	60	64	100	24000	562,539
Museum	5	70	93	100	42000	1,256,054

were used in various chapters throughout this thesis. Additional simple test graphs are shown in Figure C.1.

The constraints described above allow for multiple searchers to move in one time step, assume that the evader resides on the node, allow for searchers to teleport between nodes, allow for non-monotonic search schedules, and allow for disconnected cleared sets. A number of these restrictions could be changed by adding additional clauses to the SAT formulation.

Table C.2: Results with MiniSAT on several test maps. "S" denotes a solution was found. "U" denotes the solver returned unsatisfiable. OOM means that the solver ran out of memory (2 GB).

Map	1 searcher	2 searchers	3 searchers	4 searchers
House	U (0.004 sec)	S (0.004 sec)	-	-
Loops	U (0.004 sec)	U (0.052 sec)	S (0.02 sec)	-
Tree/grid	U (0.02 sec)	U (0.088 sec)	U (0.548 sec)	S (0.396 sec)
Gates	U (0.028 sec)	U (0.9761 sec)	S (38.1904 sec)	-
Cave	U (0.416 sec)	U (47.387 sec)	S (3743 sec)	-
NSH	U (0.268 sec)	U (355.242 sec)	OOM ($>$ 20 hours)	-
Museum	U (0.8601 sec)	U (25.0056 sec)	OOM ($>$ 20 hours)	OOM ($>$ 20 hours)

C.3 SAT Experiments

The SAT reduction was tested using the MiniSat solver (http://minisat.se/) on graphs of increasing complexity (see Table C.2). The results suggest the following:

1. The difficulty of solving a SAT reduction of a graph search problem appears to scale more with the size of the graph than with the search number of the graph. The four-searchable Tree/Grid graph was nearly as easy to solve as the three-searchable Loops graph of comparable size. The Gates map was relatively easy to solve relative to the Cave map, even though they are both three-searchable. This is likely because the Gates map contains few cycles relative to the Cave graph. The larger maps, Museum and NSH, caused the solver to run out of memory due to the large number of clauses, even though NSH is three-searchable.

2. The number of clauses in this reduction grows too quickly for MiniSAT to solve large instances of the graph search problem. However, approximate graph search techniques, such as GSST (see Chapter 4), easily find clearing schedules on the NSH and Museum maps. For SAT reductions to be competitive with algorithms like GSST, it would be necessary to develop a more compact reduction or to utilize domain-specific SAT solvers.

The SAT reduction can be used to provide a lower bound on the search number. If an unsatisfiable result is returned by the solver, then a clearing schedule is not possible with that number of searchers. This lower bound on the search number can augment the upper bound found by solvers like GSST to narrow down the search number for large graphs. For instance, the results in Table C.2 coupled with the three-searcher clearing schedule found by GSST prove that the search number of the NSH map is three.