Efficient, Guaranteed Search with Multi-Agent Teams

Geoffrey Hollinger and Sanjiv Singh
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
Email: {gholling, ssingh}@ri.cmu.edu

Athanasios Kehagias
Division of Mathematics
Aristotle University of Thessaloniki
Thessaloniki, Greece GR54124
Email: kehagiat@gen.auth.gr

***Abstract*— Here we present an anytime algorithm for clearing an environment using multiple searchers. Prior methods in the literature 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). We introduce an algorithm that combines finite-horizon planning with spanning tree traversal methods to generate plans that clear the environment of a worst-case adversarial target *and* have good average-case performance considering a target motion model. Our algorithm is scalable to large teams of searchers and yields theoretically bounded average-case performance. We have tested our proposed algorithm through a large number of experiments in simulation and with a team of robot and human searchers in an office building. Our combined search algorithm both clears the environment and reduces average capture times by up to 75% when compared to a purely worst-case approach.**

I. Introduction

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. Such a scenario may occur in urban search and rescue [1], military operations, network decontamination [2], or even aged care [3]. 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.

Known algorithms 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 and also guarantees finding a target if the model is wrong.

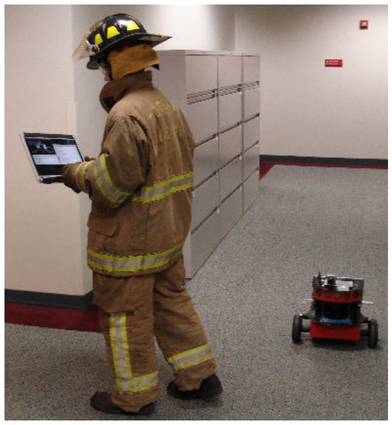
Fig. 1. Volunteer firefighter searching with a Pioneer mobile robot. The robot and humans exectue a combined schedule that both clears the environment of an adversarial target and optimizes a non-adversarial target motion model. Our decentralized algorithm allows for robots and humans flexibly to fill the different search roles. The agents share their paths and an estimation of the target's position.

In this paper, we bridge the gap between worst-case (or guaranteed) search and average-case (or efficient) search. We propose a novel algorithm that augments a guaranteed clearing schedule with an efficient component based on a non-adversarial target motion model. We extend a guaranteed search spanning tree traversal algorithm to optimize clearing time, and we augment it with a decentralized finite-horizon planning method in which agents implicitly coordinate by sharing plans. We show that the average-case performance of the combined algorithm is bounded, and we demonstrate how our algorithm can be used in an anytime fashion by providing additional search schedules with increasing runtime. This produces a family of clearing schedules that can easily be selected before the search or as new information becomes available during the search. The contribution of this paper is the first algorithm to provide guaranteed solutions to the clearing problem and additionally improve the performance of the search with fortuitous information.

The remainder of this paper is organized as follows. We first discuss related work in both worst-case search and average-case search highlighting the lack of a combined treatment

(Section II). We then define both the worst-case and average-case search problems and show the formal connection between the two (Section III). This leads us to the presentation of our search algorithm, including a description of the finite-horizon and spanning tree traversal components, as well as theoretical analysis of performance bounds (Sections IV and V). We then test our algorithm through simulated trials and through experiments on a heterogeneous human/robot search team (Section VI). Finally, we conclude and discuss avenues for future work (Section VII).

II. Related Work

As mentioned above, literature in autonomous search can be partitioned into average-case and worst-case problems. The initial formulation of the worst-case graph search problem is due to Parsons [4]. He described a scenario where a team of agents searches for an omniscient, adversarial evader with unbounded speed in a cave-like topology represented by a graph. In this formulation, the edges of the graph represent passages in the cave, and the nodes represent intersections. The evader hides in the edges (passages) of the graph, and it can only move through nodes that are not guarded by searchers. This problem was later referred to as *edge search* since the evader hides in the edges of the graph. Parsons defined the edge search number of a graph as the number of searchers needed to guarantee the capture of an evader on that graph. Megiddo et al. later showed that finding the edge search number is NP-complete for arbitrary graphs [5].

Several variations of the edge search problem appear in the literature that place restrictions on the form of the *cleared set*: the nodes on the graph that have been cleared of a potential target. *Connected edge search* requires that the cleared set be a connected subgraph at all times during the search. Barrière et al. argued that connected edge search is important for network decontamination problems because decontaminating agents should not traverse dangerous contaminated parts of the graph. They also formulated a linear-time algorithm that generates search schedules with the minimal number of searchers on trees [2].

Unfortunately, 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.[1] For this reason, robotics researchers have studied alternative formulations of the guaranteed search problem. LaValle et al. formulated the search problem in polygonal environments and presented a complete algorithm for clearing with a single searcher [6]. However, their algorithm shows poor scalability to large teams and complex environments. Gerkey et al. demonstrate how a stochastic optimization algorithm (PARISH) can be used to coordinate multiple robotic searchers in small indoor environments [7]. Though it is more scalable than complete algorithms, PARISH still requires considerable computation and communication between searchers, which makes it difficult to apply to complex environments.

In our prior work, we examined the *node search* problem in which the evader hides in the nodes of a graph, and we showed how search problems in the physical world can be represented as node search.[2] We proposed the Guaranteed Search with Spanning Trees (GSST) anytime algorithm that finds connected node search clearing schedules with a near-minimal number of searchers [9]. GSST is the first anytime algorithm for guaranteed search, and it is linearly scalable in the number of nodes in the graph, which makes it scalable to large teams and complex environments. Note that the algorithms described above (including our own prior work) do not optimize the time to clear the environment, and they do not take into account a model of the target's movement beyond a worst-case model.

A different, but closely related, search problem arises if we relax the need to deal with an adversarial target. If the target's motion model is non-adversarial and approximately known to the searchers, the searchers can optimize the average-case performance of their search schedule given this motion model. Assuming a Markovian motion model (i.e., the target's next position is dependent only on the target's current position), the average-case search problem can be expressed as a Partially Observable Markov Decision Process (POMDP). Near-optimal solutions to fairly large POMDPs are possible [10]; however, the size of search problems scales exponentially in the number of searchers. Thus, search problems with even moderately sized teams are outside the scope of general POMDP solvers.

In response to the intractability of optimal solutions, researchers have proposed several approximation algorithms for average-case search. Sarmiento et al. examined the case of a stationary target and presented a scalable heuristic [11]. Singh et al. discussed the related problem of multi-robot informative path planning and showed a scalable constant factor approximation algorithm in that domain [12]. In our prior work, we extended these approximation guarantees to average-case search with our Finite-Horizon Path Enumeration with Sequential Allocation (FHPE+SA) algorithm [13]. Our algorithm provides near-optimal performance in our test domains, but it does not consider the possibility that the model is inaccurate. Thus, the search may last for infinite time if the target is acting in a way that is not properly modeled. For instance, if a target is modeled as moving but is actually stationary, the searchers may never examine parts of the environment because they believe the target would have moved out of them.

To the best of our knowledge, no search algorithm exists that can clear an environment of a worst-case target and improve average-case search performance based on additional

[1]One exception would be a road network, which could be modeled as an edge search problem. The application of our combined algorithm to these types of environments is left for future work.

[2]Note that several alternative versions of "node search" appear in the literature (see Alspach [8] for a survey). In one formulation, the evader resides in the edges of the graph, and these edges are cleared by trapping (i.e., two searchers occupy the adjacent nodes). In another, the pursuers have knowledge of the evader's position while attempting to capture the evader by moving onto the same node.

information (e.g., a target motion model or online sensor data). Our algorithm fills this gap.

III. Problem Setup

In this section, we define the search problem with respect to both a worst-case adversarial target and a non-adversarial target. We also show the formal connection between worst-case search and the guaranteed search problem found in the literature. 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 (see Figure 2 for examples in indoor environments). At all times $t = 1, 2, \ldots, T$, searchers and a 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 (the formal definition of G' is given in the Appendix).

The searchers' movements are controlled, and they are limited to feasible paths on G'. We will refer to the path for searcher k as $A_k \subset N'$, and their combined paths as $A = A_1 \cup \ldots \cup A_K$. 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 takes path Y, the searchers receive reward $F_Y(A) = \gamma^{t_A}$, where $t_A = \min\{t : (m, t) \in A \cap Y\}$ (i.e., the first time at which path Y intersects path A), with the understanding that $\gamma \in (0, 1)$, $\min \emptyset = \infty$, and $\gamma^\infty = 0$. Thus, if the paths do not intersect, the searchers receive zero reward.

This paper considers two possible assumptions on the target's behavior, which yield the average-case and worst-case search problems. If we make a non-adversarial assumption on the target's behavior, we can utilize a target motion model independent of the locations of the searchers. This yields a probability $P(Y)$ for all possible target paths $Y \in \Psi$. We can now define the optimization problem in Equation 1. Equation 1 maximizes the *average-case* reward given a motion model defined by $P(Y)$. Note that if the target's motion model obeys the Markov property, we can estimate a probability distribution of its location efficiently using matrix algebra.

$$A^* = \underset{A}{\operatorname{argmax}} \sum_{Y \in \Psi} P(Y) F_Y(A) \tag{1}$$

The average-case optimization problem in Equation 1 does not consider the possibility that the motion model may be incorrect. An alternative assumption on the target's behavior is to assume that it actively avoids the searchers as best possible. For the search problem, this implies that the target chooses path Y that minimizes $F_Y(A)$. This yields the game theoretic optimization problem in Equation 2. Here, the searchers' goal is to maximize the *worst-case* reward if the target acts as best it can to reduce reward.

$$A^* = \underset{A}{\operatorname{argmax}} \min_{Y} F_Y(A) \tag{2}$$

It is important to note that this formulation assumes that the searchers have a "perfect" sensor that will always detect the target is it resides in the same cell. We can relax this assumption 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).

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 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.

$$A^* = \underset{A}{\operatorname{argmax}} \quad \alpha \sum_{Y \in \Psi} P(Y) F_Y(A) + (1 - \alpha) \min_{Y} F_Y(A), \tag{3}$$

where $\alpha \in [0, 1]$ is a weighting variable to be tuned based on the application.

While scalarization is a viable approach, we argue that it makes the problem more difficult to solve. Note that the underlying function $F_Y(A)$ is nondecreasing and submodular (see Appendix for definition) as is its expectation in Equation 1. In fact, Equation 1 is the efficient search (MESPP) optimization problem as defined by Hollinger and Singh [13], which can be optimized using the FHPE+SA algorithm. FHPE+SA yields a bounded approximation 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 is not bounded and, in fact, performs poorly in practice. Furthermore, Krause et al. show that game theoretic sensor placement problems of similar form to Equation 2 do not yield any bounded approximation (unless $P = NP$) [14]. Thus, scalarization combines an easier problem with a more difficult problem, which prevents exploiting the structure of the easier MESPP problem.

We propose treating the combined search problem as a resource allocation problem. More specifically, some searchers make the average-case assumption while others make the worst-case assumption. One case where this approach can improve the search schedule is in the (very common) scenario where a portion of the map must be cleared before progressing. In this scenario, several searchers are often assigned to guard locations while waiting for the other searchers to finish clearing. Some of these guards can be used to explore the uncleared portion of the map. An example of this case is shown in our human-robot experiments in Section VI.

If the searchers are properly allocated to the average-case and worst-case tasks, we can generate search schedules 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. The question now becomes: how many searchers do we assign 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 set of paths A for a minimal number of searchers K_{min} such that A is a clearing path [9, 2]. In other words, find a path that clears the environment of an adversarial evader using the smallest number of searchers. Propositions 1 and 2 show the connection between node clearing and the worst-case search problem defined in Equation 2.[3]

Proposition 1: The value $\min_Y F_Y(A) > 0$ if and only if A is a clearing path (i.e., a set of paths that clear the environment of any target within it).

Proof: Assume that A is not a clearing path and $\min_Y F_Y(A) > 0$. Since A is not a clearing path, this implies that one or more nodes in G are contaminated (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 contaminated set for all t. Such a path is feasible due to the assumptions of the evader and the recontamination rules. The contaminated 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 path and $\min_Y F_Y(A) = 0$. This assumption implies that $A \cap Y = \emptyset$. By definition of clearing path, the contaminated set of $G = \emptyset$ at some time T. However, this implies that $A \cap Y \neq \emptyset$ or else there would exist a contaminated cell. Again we have a contradiction. ■

Proposition 2: For a given number of searchers K and graph G, $\max_A \min_Y F_Y(A) = 0$ 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 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 1, this implies that a schedule A does not exist such that $\min_Y F_Y(A) > 0$ for $K < s(G)$. ■

Proposition 1 shows that any non-zero solution to the optimization problem in Equation 2 will also be a node search solution with K searchers. Proposition 2 shows that $s(G)$, the node search number of G, will affect the optimization problem in Equation 2; if $K < s(G)$, then $\min_Y F_Y(A) = 0$ for all A. Drawing on this analysis, we now know that at least $s(G)$ searchers must make the worst-case assumption to generate a schedule with any nonzero worst-case reward. This motivates 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. We show how this limitation can be overcome in the next section.

[3]Note that Propositions 1 and 2 hold for both monotone and non-monotone clearing schedules. In other words, the propositions hold regardless of whether recontamination is allowed in the schedule.

IV. Algorithm Description

Drawing off the observations in the previous section, we can design an algorithm 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 optimizes clearing time with a given number of searchers. We extend the Guaranteed Search with Spanning Trees (GSST) algorithm [9] 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. show how to generate a recursive labeling of a tree (we will refer to this labeling as B-labeling) in linear-time [2]. Informally, the labeling determines how many searchers must traverse a given edge of the tree in the optimal schedule (see Algorithm 1). 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 create cycles to generate a guaranteed search schedule on arbitrary graphs (see reference [9] for more detail). However, GSST gives no mechanism for utilizing more searchers than the minimal number. Thus, it does not optimize clearing time.

Algorithm 1 B-labeling for Trees
```
1: Input: Tree $T(N, E)$, Start node $b \in N$
2: $O \leftarrow$ leafs 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:    if $l \neq b$ and $parent(l)$ ready for labeling then
17:      $O \leftarrow O \cup parent(l)$
18:    end if
19:  end if
20: end while
21: Output: B-labeling $\lambda(E)$
```

Algorithm 2 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. Algorithm 2 does not explicitly use guards on edges creating cycles; the guards are instead

Algorithm 2 Guardless Guaranteed Search with Spanning Trees (G-GSST)

```
1: Input: Graph $G$, Spanning tree $T$ with B-labeling, Searchers $K_g$
2: while graph not cleared do
3:   for all searchers do
4:     if moving will recontaminate then
5:       Do not move
6:     else if positive adjacent edge label exists then
7:       Travel along edge with smallest positive label
8:       Decrement edge label
9:     else
10:      Move to closest node (on tree) with positive label
11:    end if
12:  end for
13:  if no moves possible without recontamination then
14:    Return failure
15:  end if
16: end while
17: Return feasible clearing schedule
```

determined implicitly from the B-labeling. Thus, we refer to it 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).

Unlike other algorithms in the literature, G-GSST can be combined with model-based search algorithms to generate a combined search schedule. We augment G-GSST with Finite-Horizon Path Enumeration and Sequential Allocation (FHPE+SA) to yield our combined search algorithm. In short, 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 reference [13] for a formal description). In other words, one 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.

Algorithm 3 shows how G-GSST can be augmented with FHPE+SA to yield a combined algorithm. An important quality of the combined search is that, depending on the actions of the FHPE+SA searchers, the schedule may be non-monotone (i.e., it may allow for recontamination). However, since the schedule of the G-GSST searchers is monotone, the search will still progress towards clearing the environment of a worst-case target.

It is important to note that the schedule of the G-GSST searchers (i.e., the searchers performing the guaranteed schedule) can be generated in conjunction with the FHPE+SA plans. For instance, we can modify the G-GSST searcher schedule based on the actions of the FHPE+SA searchers by pruning

Algorithm 3 Combined G-GSST and FHPE+SA search

```
1: Input: Graph $G$, Spanning tree $T$ with B-labeling, Total searchers $K$, Guaranteed searchers $K_g \leq K$
2: while graph not cleared do
3:   for all searchers do
4:     if guaranteed searcher then
5:       Run next step of G-GSST
6:     else
7:       Run FHPE+SA step
8:     end if
9:   end for
10: end while
```

Algorithm 4 Anytime combined search algorithm

```
1: Input: Graph $G$, Searchers $K$
2: while time left do
3:   Generate spanning tree $T$ of $G$ and B-label it
4:   for $K_g = K$ down to $K_g = 1$ do
5:     Run Algorithm 3 with $K_g$ guaranteed searchers and $K - K_g$ efficient searchers
6:     if clearing feasible then
7:       Store strategy
8:     else
9:       Break
10:    end if
11:  end for
12: end while
13: if strategies stored then
14:   Return strategy with maximal $\alpha R_{avg} + (1 - \alpha) R_{worst}$
15: else
16:   Run FHPE+SA to maximize $R_{avg}$ (cannot clear)
17: end if
```

portions of the map that happen to be cleared by the FHPE+SA searchers.[4] This provides a tighter coupling between the FHPE+SA searchers and the G-GSST searchers. Alternative methods for integrating the search plans are discussed in Section VII.

Finally, we generate many spanning trees (similarly to the GSST algorithm) to develop many search schedules (see Algorithm 4). 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/her needs. This yields an anytime solution to the worst-case/average-case search problems: one that continues to generate solutions with increasing runtime.

[4]Note that implementing this extension requires ensuring that ignoring a portion of the map cleared by the FHPE+SA searchers does not lead to later recontamination. This can be determined by dynamically relabeling the spanning tree and taking into account the cleared set. Since labeling is a linear-time operation, this does not significantly affect runtime.

V. Theoretical Analysis

A. Average-Case Performance Bounds

We now show that the average-case component of the combined algorithm is a bounded approximation. Let A be the set of nodes visited by the FHPE+SA algorithm. Let O be the set of nodes visited by the optimal average-case path (maximizes Equation 1), and let O_k be the set of nodes visited by searcher k on the optimal path. The average-case reward is bounded as in Theorem 1. This bound is simply the FHPE+SA bound for $K - K_g$ searchers.

Theorem 1:

$$F_{AC}(A) \geq \frac{F_{AC}(O_1 \cup \ldots \cup O_{K-K_g}) - \epsilon}{2}, \quad (4)$$

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). $F_{AC}(\cdot) = \sum_{Y \in \Psi} P(Y) F_Y(\cdot)$ as described in Section III.

Proof: This bound is immediate from the theoretical bounds on sequential allocation [12], the monotonic submodularity of F_{AC} [13], and the fact that $K - K_g$ searchers are deployed to perform sequential allocation. The addition of G-GSST searchers cannot decrease $F_{AC}(A)$ due to the monotonicity of F_{AC} and the monotonicity of the cleared set in the G-GSST schedule. ■

We can extend the FHPE+SA component of the combined algorithm to optimize over several different known models. If there are M models being considered, and each has a probability of $\beta_1 \ldots \beta_M$, we can optimize the weighted average-case objective function in Equation 5.

$$F(A) = \beta_1 \sum_{Y \in \Psi} P_1(Y) F_Y(A) + \ldots + \beta_M \sum_{Y \in \Psi} P_M(Y) F_Y(A), \quad (5)$$

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 without an increase in computation. Additionally, monotonic submodularity is closed under non-negative linear combination, so the theoretical bounds hold in this extended case.

B. Computational Complexity

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 average 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 often can be tuned to a very low number depending on the application.

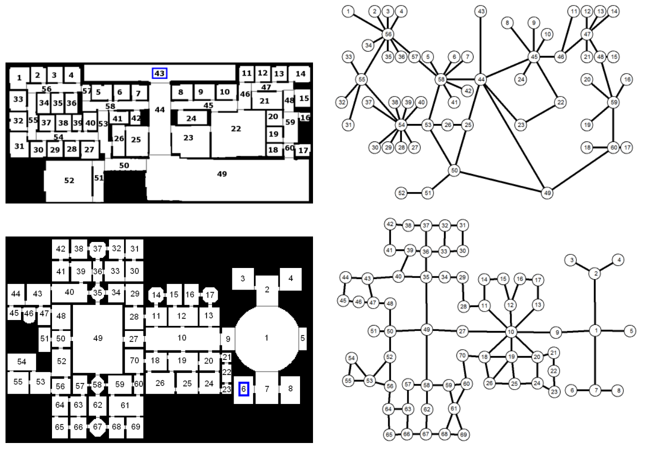
Fig. 2. Example floorplans (left) and graphical representations (right) of environments used for simulated coordinated search. The office (top) required three searchers to clear, and the museum (bottom) required five. Starting cells are denoted with a blue square.

VI. Experimental Results

A. Simulated Results

We performed simulated testing of our combined search algorithm in two complex environments. The first map has two major cycles representing the hallways in a single floor of the Newell-Simon Hall office building. The second map is that of the U.S. National Gallery museum, which contains many cycles. Increasing the number of cycles in a graph complicates the search problem because it increases the paths by which the target can avoid being found. Figure 2 shows floorplans of these environments as well as the resulting search graphs.

We ran our algorithm on 1000 random spanning trees on both maps, and we compared these results to the schedules found by pure average-case and pure worst-case algorithms. Table I gives a summary of these results. The first row shows the average-case steps (i.e., at each step one or more searchers moves between nodes) to capture a randomly moving target using FHPE+SA with all searchers.[5] 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 our combined algorithm yields much lower clearing times than those found with GSST. This is due to the use of additional searchers as in Algorithm 2. In addition, our combined algorithm reduces the average capture time by over 75% in the office when compared to GSST. Furthermore, a worst-case guarantee in the office can be gained by sacrificing only one step of average capture time.

[5]The expected capture steps for FHPE+SA were determined over 200 trials with a randomly moving target. The expected capture steps for the combined algorithm were computed in closed form. This is possible because the environment is cleared.

TABLE I

AVERAGE-CASE AND WORST-CASE CAPTURE STEPS COMPARISON IN MUSEUM AND OFFICE. THE AVERAGE-CASE IS THE EXPECTED STEPS TO CAPTURE A RANDOMLY MOVING TARGET. THE WORST-CASE IS THE NUMBER OF STEPS TO CLEAR THE ENVIRONMENT.

	Office ($K = 5$)	Museum ($K = 7$)
FHPE+SA [13]	A.C. 4.8 W.C. ∞	A.C. 9.4 W.C. ∞
GSST [9]	A.C. 24.7 W.C. 72	A.C. 21.8 W.C. 56
Combined ($\alpha = 0.5$)	A.C. 14.0 W.C. 36	A.C. 17.3 W.C. 41
Combined ($\alpha = 0.75$)	A.C. 7.2 W.C. 47	A.C. 13.5 W.C. 45
Combined ($\alpha = 0.99$)	A.C. 5.9 W.C. 77	A.C. 11.6 W.C. 61

The user can determine the weighting of average-case vs. worst-case performance by tuning the value of the α parameter. This explores the Pareto-optimal frontier of solutions. This frontier forms a convex hull of solutions, any of which can be selected after runtime. Figure 3 gives a scatter plot of average-case versus worst-case capture steps for all feasible schedules. The figure also shows the number of G-GSST searchers, 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 performed FHPE+SA. All results with lowest average-case capture times are from the lowest K_g. This is because more searchers are used for average-case search and less for clearing. This demonstrates the utility of using FHPE+SA searchers in the schedule. Similarly, the lowest clearing times are with $K_g = K$ (i.e., all searchers are guaranteed searchers). Note that better solutions yield points to the left/bottom of these plots.

B. Human-Robot Teams

To examine the feasibility of our algorithm, we ran several experiments with a human/robot search team. We conducted these experiments on a single floor of an office building as shown in Figure 4. 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 3. The robot's position is determined by laser AMCL, and it is given waypoints through the Player/Stage software [15]. The humans input their position through the keyboard, and they are given waypoints through a GUI.

In the first experiment, all three searchers (humans and robots) were assigned as G-GSST searchers. This configuration takes 178 seconds to clear the floor of a potential adversary, and it yields an expected capture time of 78 seconds w.r.t. the random model. We also calculated the expected capture time if the model is 50% off (i.e., the target is moving 50% slower than expected). With this modification, the expected capture time increases to 83 seconds.

In the second experiment, the mobile robot was switched to an FHPE searcher, which 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 FHPE searcher 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 shows that the schedules generated in this scenario are fairly robust to changes in the target's motion model.

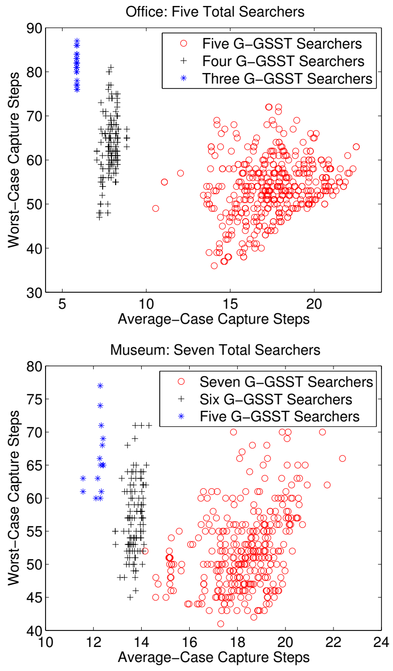

Fig. 3. Scatter plot of search schedules from the combined algorithm. Each data point represents a schedule generated by a different spanning tree input to Algorithm 3. The data points are colored based on the allocation of searchers to the guaranteed and efficient search roles. The total searchers remains constant throughout the trials.

In both experiments, the schedule clears the left portion of the map first and then continues to clear the right portion. Accompanying videos are available at the following URL that include playback of both simulated and human-robot results.

`http://www.andrew.cmu.edu/user/gholling/RSS09/`

In this environment, the assignment of the robot as an FHPE searcher does not increase the clearing time. The reason for this 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 FHPE searcher, the right hallway is searched early in the schedule, which would locate a non-adversarial target moving in that area. This confirms our simulated results.

In addition, our results with a human/robot team demonstrate the feasibility of the communication and computational requirements of our 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 search graph. This small amount of information was easily handled by a standard wireless network in an academic building.

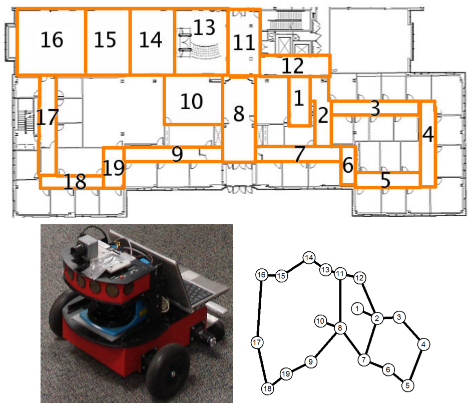

Fig. 4. Map of office building (top) used for experiments with a human/robot search team, and corresponding search graph (bottom right). Pioneer robot searcher (bottom left) with laser rangefinder and camera. The robot used the laser rangefinder for localization, and it used the camera to provide video feed to the humans. The robot and human team started at the entrance in cell 8.

VII. Conclusion and Future Work

This paper has presented an algorithm for generating multi-agent search paths that both clear an environment of a potential adversary *and* optimize over a non-adversarial target motion model. Our algorithm bridges the gap between guaranteed search and efficient search by providing a combined search algorithm with both worst-case and average-case guarantees. We have shown through simulated experiments that our algorithm performs well when compared to state-of-the-art search algorithms in both guaranteed and efficient search. In addition, we have demonstrated the feasibility of our algorithm on a heterogeneous human/robot search team in an office environment.

Our current algorithm does not directly use a weighting variable α to incorporate confidence in the model. Instead, we cache many solutions and allow 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.

Additional future work includes more extensive robustness testing with inaccurate motion models and analysis of the communication requirements with very large search teams. Future field testing includes applications in emergency response, military reconnaissance, and aged care. Combining search guarantees against an adversarial target with efficient performance using a model has the potential to improve autonomous search across this wide range of applications. This paper has contributed the first algorithm and results towards this goal.

Acknowledgment

The authors gratefully acknowledge Joseph Djugash, Ben Grocholsky, and Debadeepta Dey from CMU for their insightful comments and assistance with experiments. We thank volunteer firefighter (and professional roboticist) Seth Koterba for feedback on the system.

References

[1] V. Kumar, D. Rus, and S. Singh, "Robot and Sensor Networks for First Responders," *Pervasive Computing*, pp. 24–33, 2004.

[2] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro, "Capture of an Intruder by Mobile Agents," in *Proc. 14th ACM Symp. Parallel Algorithms and Architectures*, 2002, pp. 200–209.

[3] N. Roy, G. Gordon, and S. Thrun, "Planning under Uncertainty for Reliable Health Care Robotics," in *Proc. Int'l Conf. on Field and Service Robotics*, 2003.

[4] T. Parsons, "Pursuit-Evasion in a Graph," in *Theory and Applications of Graphs*, Y. Alavi and D. Lick, Eds. Springer, 1976, pp. 426–441.

[5] N. Megiddo, S. Hakimi, M. Garey, D. Johnson, and C. Papadimitriou, "The Complexity of Searching a Graph," *Journal of the ACM*, vol. 35, pp. 18–44, 1988.

[6] L. Guibas, J. Latombe, S. LaValle, D. Lin, and R. Motwani, "Visibility-Based Pursuit-Evasion in a Polygonal Environment," *Int'l Journal of Comp. Geometry and Applications*, vol. 9, no. 5, pp. 471–494, 1999.

[7] B. Gerkey, S. Thrun, and G. Gordon, "Parallel Stochastic Hill-Climbing with Small Teams," in *Proc. 3rd Int'l NRL Workshop on Multi-Robot Systems*, 2005.

[8] B. Alspach, "Searching and Sweeping Graphs: A Brief Survey," *Matematiche*, pp. 5–37, 2006.

[9] G. Hollinger, A. Kehagias, S. Singh, D. Ferguson, and S. Srinivasa, "Anytime Guaranteed Search using Spanning Trees," Robotics Institute, Carnegie Mellon University, CMU-RI-TR-08-36, Tech. Rep., 2008.

[10] T. Smith, "Probabilistic Planning for Robotic Exploration," Ph.D. dissertation, Carnegie Mellon University, 2007.

[11] A. Sarmiento, R. Murrieta-Cid, and S. Hutchinson, "A Multi-Robot Strategy for Rapidly Searching a Polygonal Environment," in *Proc. 9th Ibero-American Conference on Artificial Intelligence*, 2004.

[12] A. Singh, A. Krause, C. Guestrin, W. Kaiser, and M. Batalin, "Efficient Planning of Informative Paths for Multiple Robots," in *Proc. Int'l Joint Conf. on Artificial Intelligence*, 2007.

[13] G. Hollinger and S. Singh, "Proofs and Experiments in Scalable, Near-Optimal Search with Multiple Robots," in *Proc. Robotics: Science and Systems Conf.*, 2008.

[14] A. Krause, B. McMahan, C. Guestrin, and A. Gupta, "Selecting Observations Against Adversarial Objectives," in *Proc. Neural Information Processing Systems*, 2007.

[15] B. Gerkey, R. Vaughan, and A. Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," in *Proc. Int'l Conf. on Advanced Robotics*, 2003, pp. 317–323.

Appendix: Definitions

Definition 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.

Definition 2: A function $F : \mathbf{P}(N') \to \Re_0^+$ is called *nondecreasing* iff for all $A, B \in \mathbf{P}(N')$, we have

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

Definition 3: A function $F : \mathbf{P}(N') \to \Re_0^+$ is called *submodular* iff for all $A, B \in \mathbf{P}(N')$ and all *singletons* $C = \{(m,t)\} \in \mathbf{P}(N')$, we have

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