Heterogeneous Exploration and Monitoring with Online Free-Space Ellipsoid Graphs

Brennan Brodt and Alyssa Pierson

***Abstract*— This paper proposes a heterogeneous teaming solution to the problem of target discovery and monitoring in unknown, non-convex environments. The team consists of two types of agents: agile agents with sensors capable of mapping their surroundings and slower agents that are capable of monitoring or servicing discovered targets. We propose an exploration algorithm that utilizes the IRIS algorithm to generate a graph decomposition from collision free ellipses contained within the environment. This graph is passed to the monitoring agents who execute polynomial complexity assignment and touring algorithms to generate high quality path plans which service all discovered targets. Our algorithmic structure allows the team to solve the problems of exploration, target discovery, assignment, and monitoring within unknown, non-convex environments efficiently using limited information. The performance of our proposed method is verified through batch simulations and complexity analysis.**

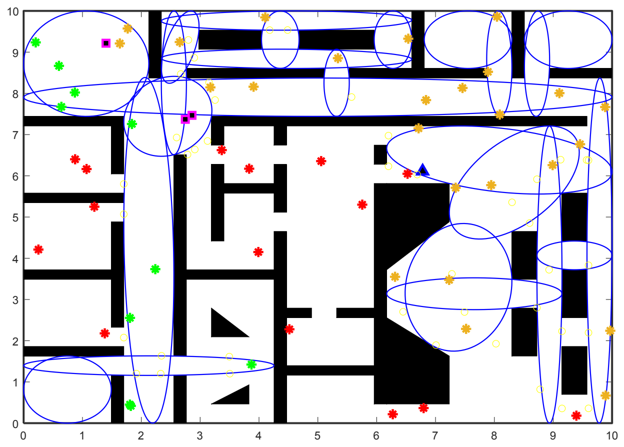


Fig. 1: Depiction of the exploration and monitoring problem being solved. Our exploring agent (blue triangle) travels through the environment to generate the ellipsoid graph and discover targets. Our monitoring agents (pink squares) plan trajectories that visit all discovered targets (orange stars), and service them (green stars).

I. Introduction

Heterogeneous teams of autonomous agents are capable of coordinating their abilities to accomplish complex mission objectives. As robots continue to expand their real world applications, scenarios of different robots collaborating on tasks will continue to become more frequent. However, designing control frameworks that are capable of harnessing these different capabilities in synergistic ways remains very challenging. Overcoming this obstacle is a long standing area of research in robotics that is currently burgeoning in the state-of-the-art [1], [2], [3], [4]. These heterogeneous teams have an incredibly wide array of applications, from firefighting and search and rescue to wildlife monitoring and delivery routing [5], [6], [7], [8], [9]. Many current methods for solving these heterogeneous teaming problems rely on large-scale, offline optimizations [4], [7] or complete environmental knowledge [10], [11] to generate plans for the team. While these produce optimal results, they require a large amount of a priori information that is unavailable in real world applications. Other methods simplify the problem by reducing team heterogeneity to a measure on performance, ability, or similar metric and generating control policies based on these [12], [13]. These are capable of producing robust closed-loop policies that perform well in coordinating agent behaviors according to their abilities but are not suited for applications where agents have unique objectives.

In this paper, we propose a heterogeneous teaming formulation for the problem of exploration and monitoring that enables a team to traverse unknown environments, construct safe graph decompositions online, and use these decompositions for high-level coordination. In this regard, our team consists of two distinct agent types: agile agents with advanced environmental sensing that carry out exploration and graph generation, and slower agents equipped with the required capabilities to service tasks discovered within the environment. This type of team formulation has many potential real world applications, such as in firefighting or search and rescue where agile UAVs can map out a disaster site while identifying remaining fires or survivors, which can then be put out or assisted by more capable UGVs. Through this collaboration, we enable a team to efficiently accomplish complex objectives in completely unknown environments with lightweight, online computations. The key contributions of this work are as follows:

1) A heterogeneous team structure for discovery and monitoring of targets in unknown environments;
2) An IRIS-inspired exploration algorithm that generates an ellipsoidal decomposition and corresponding graph online for high-level team coordination; and
3) A polynomial complexity algorithm that optimally assigns targets to monitors and generates bounded approximations of the associated optimal TSP tours.

The remainder of this paper is organized as follows: Section II presents related work. Section III formalizes the problem formulation. Sections IV and V describe our algorithmic solutions. Section VI analyzes our simulation results. Finally, Section VII states our conclusions.

Brennan Brodt and Alyssa Pierson are with the Department of Mechanical Engineering, Boston University, Boston, MA 02215, USA `[brodt, pierson]@bu.edu`.

II. Related Work

Our work in this paper builds on a wide array of foundational and contemporary robotics research. At its core, this paper presents a take on heterogeneous teaming for the problems of exploration and monitoring. Here, heterogeneous teaming refers to the joint collaboration of robots with different capabilities on a singular team objective. Exploration refers to the problem of traversing unknown environments in search of environmental obstacles and objectives. Finally, monitoring refers to the problem of travelling between key locations within the environment to provide services or take measurements.

Heterogeneous Teaming: As robot manufacturers continue to proliferate specialized systems to industrial and commercial applications, the ability to coordinate mixed fleets will only become more important. Research on this problem is currently very active as mixed fleets can improve team performance through the coordination of specialized agents [2], [6], [7]. Some common applications for heterogeneous teaming include multistage exploration and recharging, arising from power limitations in aerial systems [11], [14]. Other metrics for heterogeneity use robot performance [12], [13] or the distinct goods and services that can be offered by each robot [10], [15], [16]. Research further in this topic goes as far as optimizing the choice of available robots for mixed fleet applications [17], [18]. In this work, we consider heterogeneity in a similar sense to [4], [5], [19], [20], where our team consists of robots with different tasks that, when combined, complete a singular team objective.

Exploration and Environmental Decomposition: Traversing unknown environments poses many difficulties to autonomous systems and planning [21]. Environmental decomposition provides a method to overcome many of these difficulties. In this setting, agents explore an environment and map the free space to a smaller graph representation containing simple rules in description, graph adjacency, and edge weights [22], [23]. Environmental decomposition has seen active development from the earliest of robotics applications [24], [25], [26], [27], [28], [29] and continues to stand on the forefront of robotics research. In this paper, we explore the IRIS decomposition method proposed in [30]. Decompositions generated by the IRIS algorithm are quick to compute and any planning performed on the decomposition is guaranteed to be collision free. Further they have been shown to perform well in conjunction with complicated dynamic constraints [31].

Target Assignment and Monitoring: Autonomous target assignment is a foundational problem in the field of robotics. Drawing heavily from identical problems in other fields [8], [32], [33], [34], [35], it is often critical that autonomous systems be capable of determining where their capabilities are best suited when acting in a team. Simple assignment algorithms are often utilized due to their high efficiency and satisfactory assignment results [36], though advanced assignment behaviors can be difficult to encode in these algorithms [3]. Modern target assignment problems often exploit heuristics [9], [15] or well understood problem structures [37], [38] to dramatically improve performance. In this work, we frame target assignment as a minimum cost maximum flow problem [39], [40]. In doing so, we can optimally distribute targets to monitoring agents and express advanced assignment behaviors by modifying the flow network, represented as a graph.

III. Problem Formulation

Our goal is to service a set of targets contained within an unknown, non-convex environment using a heterogeneous team of autonomous agents. To accomplish the team objective, we split the problem into two subtasks: exploration and monitoring — corresponding to the two agent types in our team. For the agile exploring agents, we seek to explore the environment, locate all targets in need of service, and generate a lightweight, collision free graph decomposition of the free space for planning by the monitors. For the monitoring agents, we seek to assign discovered targets to available monitors and generate feasible TSP tours of assigned targets on the graph decomposition reported by the explorers. We make five key assumptions:

1) Our heterogeneous team consists of two distinct types of agents with global communications;
2) Exploring agents are capable of fully sensing and localizing within their surrounding environment and can discover any target within line of sight;
3) Monitoring agents can service any discovered target but are only capable of localizing on the graph generated by the explorers;
4) Inter-robot collisions are avoided via low level control;
5) The environment is bounded and obstacles are described as combinations of convex polygons.

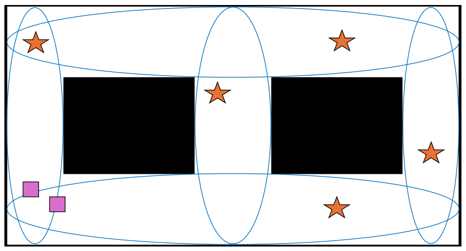
Given these assumptions, consider a bounded environment $Q \subset \mathcal{R}^2$ containing a set of convex obstacles $\mathcal{O} \subset Q$. Within this environment are a set of random target positions, given by $\mathcal{T} = [t_1, t_2, ..., t_n] | t_i \in Q \backslash \mathcal{O}$. We denote $p_m(t), p_e(t) \in Q \backslash \mathcal{O}$ as the positions of our monitors and explorers, respectively. All agents are assumed to have integrator dynamics $\dot{p}_m(t), \dot{p}_e(t) = u$, where the desired control input u is bounded by some $|u| \leq u_{max,m} \leq u_{max,e}$.

Then our exploring agents are tasked with generating a graph $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{T}_d)$, where $\mathcal{V}$ are the graph vertices, $\mathcal{E}$ are the graph edges, and $\mathcal{T}_d \subset \mathcal{T}$ are the discovered targets. Ideally, explorers generate $\mathcal{G}$ such that $\mathcal{T}_d = \mathcal{T}$ and $Q \backslash (\mathcal{O} \cup \mathcal{V})$ is minimized, while also maintaining $\mathcal{O} \cap \mathcal{V} = \emptyset$. Intuitively, these rules mean an ideal graph decomposition contains all targets within the environment, maximizes coverage over the free space, and contains no collisions with obstacles.

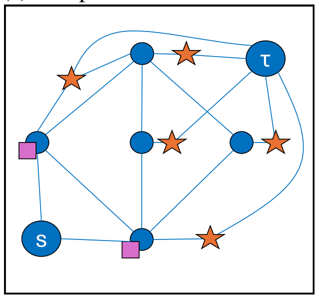
Our monitoring agents are tasked with optimally assigning the targets in $\mathcal{T}_d$ to themselves and generating a sequence $\mathcal{S} \in \mathcal{V} \cup \mathcal{T}_d$, such that $\mathcal{T}_d \subset \mathcal{S}$. Intuitively, all targets that have been discovered by our exploring agents need to be assigned to a monitor, and the generated sequence must exist completely within the vertices of the graph and discovered target positions while also ensuring that all discovered targets are serviced eventually.

IV. Exploration and Graph Generation

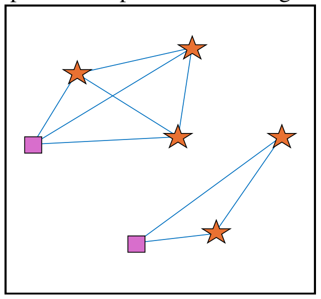
In Section III we described the overarching problem setting for the entire team. Here, we introduce an algorithmic



(a) Sample Environment With Ellipse Decomposition and Targets



(b) Assignment Graph



(c) Touring Graph

Fig. 2: (a) Depicts a simple environment containing monitors, pink squares, and targets, orange stars. Resulting graphs for algorithms 2 and 3 are depicted in (b) and (c), respectively.

solution to the exploring agents' subtask. Our goal is to generate a decomposition of the environment, computed online during exploration, that is represented with a graph and can be used for collision free path planning. We build upon the author's prior work on Free-Space Ellipsoid Graphs [31], which aided in planning across multi-robot teams in non-convex environments. In this work, we aim to use the IRIS algorithm to generate a decomposition online, as opposed to previous methods which rely on offline sampling. While random sampling generally ensures coverage, it requires full environmental knowledge and often leads to overlapping and redundant vertices. As described in [30], the IRIS algorithm operates by identifying a collision free convex hull around a given seed point and inflating the maximum volume ellipsoid within the hull. The results of this process are returned in the form of linear constraints such that:

$$[Aq \leq b \,|\, q \in \mathcal{Q}] \cap \mathcal{O} = \emptyset, \tag{1}$$

where matrix A and vector b define a set of hyperplanes separating the convex hull from all obstacles within the environment; as well as an ellipse, defined as:

$$C\tilde{q} + d \,|\, \|\tilde{q}\| \leq 1, \tag{2}$$

such that matrix C and vector d define a dilation and translation of the unit ball to generate the maximal volume ellipse contained within the convex hull defined by $[A, b]$.

By running the IRIS algorithm from multiple different seed points, we can quickly construct a graph from the ellipsoidal representation by defining the ellipses as vertices in the graph, with edges determined by any intersections between them. Further, we can include targets in this graph by connecting them to the ellipses associated with the convex hulls in which they are contained. An example of this decomposition is depicted in Figure 2. Given a connected ellipsoid graph

$$\mathcal{G}(\mathcal{V}_i = [A, b, C, d], [\mathcal{V}_i \cap \mathcal{V}_j], \mathcal{T}_d),$$

Propositions 1 and 2 show that there exists a collision free trajectory to any $\mathcal{T}_d$ within $\mathcal{G}$.

Proposition 1. *There exists a collision free trajectory from any point within a convex hull to the corresponding ellipse.*

Proof. Convex hulls generated by the IRIS algorithms are collision free, given by (1). Further, any point $\bar{q}$ that satisfies:

$$C^{-1}(\bar{q} - d) \leq 1, \tag{3}$$

also satisfies:

$$A\bar{q} \leq b, \tag{4}$$

as generated ellipses are fully contained within their convex hulls. Therefore, any path from ellipse to convex hull is contained within the convex hull and must be collision free. □

Corollary 1. *Any target discovered within a convex hull can be reached with a collision free trajectory from the corresponding ellipse.*

Proof. Discovered targets are marked at points within generated convex hulls. Following Proposition 1, a path to any discovered target is fully contained within the convex hull and is therefore collision free. □

Proposition 2. *There exists a collision free trajectory between any point in an ellipse to at least one point in an adjacent ellipse.*

Proof. Consider any two adjacent ellipses, given by $[C_1, d_1]$ and $[C_2, d_2]$. There exists some $\bar{q}$ such that:

$$C_1^{-1}(\bar{q} - d_1) = C_2^{-1}(\bar{q} - d_2) \text{ for } \bar{q} \in \mathcal{Q}, \tag{5}$$

which implies $A_1\bar{q} \leq b_1$. Since the convex hull $[A_1, b_1]$ that generates $[C_1, d_1]$ is collision free, there must exist a collision free straight line that travels from any point in $[C_1, d_1]$ to point $\bar{q}$ in $[C_2, d_2]$. □

Corollary 2. *There exists a collision free trajectory from any point within a convex hull to at least one point in an adjacent ellipse.*

Proof. Following Proposition 2, at least one point in an adjacent ellipse must also be in the current ellipse. The current ellipse is fully contained within the current convex hull. By Proposition 1, there must exist a path to a point in an adjacent ellipse that is fully contained within the convex hull and therefore collision free. □

These volumes are useful tools for the exploration and mapping of environments because the convex hull allows agents to flag potential frontiers for exploration and any planning performed on the ellipse representation is guaranteed to be safe. Further, from the standpoint of scalability and communications, the generated ellipses are fully described through C and d which are consistent in size, regardless of their orientation within the environment. To use these

Algorithm 1 Exploration and Graph Generation

```
Input: Explorer Position: $p_e(t)$, Stack: $\mathcal{S}$, Graph: $\mathcal{G}$
Init: Stack $\mathcal{S} \leftarrow p_e(t)$
while $\mathcal{S}$ is not empty do
    Set waypoint $w$ to top of $\mathcal{S}$
    if Explorer at waypoint: $p_e(t) = w$ then
        Remove top element from stack $\mathcal{S}$
        Run IRIS: $A, b, C, d \leftarrow$ inflate($w$)            ▷ [30]
        if $C, d$ already exist in $\mathcal{V}$ then
            Move all flags of matched vertex to top of $\mathcal{S}$
        else
            Add new vertex, $[C, d]$, to $\mathcal{G}$
            Compute intersections between $[C, d]$
                and all other ellipses in $\mathcal{V}$
            Update $\mathcal{E}$ per existence of intersections
            Generate convex hull using $A, b$
            Add any $t$ in convex hull to $\mathcal{T}_d$
                Note: position and vertex index
            Check convex hull for collision with $\mathcal{O}$
            Flag midpoint of each collision free segment
            Add all flags and vertex indices to top of $\mathcal{S}$
    else
        if $w$ is in current vertex then
            Move $p_e(t)$ towards $w$
        else
            Run Dijkstra from current vertex            ▷ [41]
            Move $p_e(t)$ towards next ellipse intersection
                along shortest path to $w$
```

results as a tool for exploration, agents need only determine where to place seed points for the IRIS algorithm such that the exploration frontier is expanded or previously uncovered spaces are filled.

Following Algorithm 1, explorers begin by placing a seed point at their initial position. This is expanded using IRIS to generate a collision free convex hull around the explorer. Any targets contained within the computed convex hull are added to $\mathcal{T}_d$. Agents then perform a line search along each edge of the convex hull to determine where they depart from nearby obstacles. We use these departures to place new flags along the convex hull that offer frontiers for further exploration. Each flag is added to a stack containing their positions and associated graph vertices. The top entry in each agent's stack is then selected as the next waypoint. Explorers utilize Dijkstra's algorithm to plan the shortest path through the graph to the vertex associated with this waypoint [41]. Upon reaching the waypoint, agents expand a new seed with IRIS, and the process of placing flags and travelling towards waypoints repeats until the stack is empty. When expanding seed points with IRIS, explorers also check the resulting ellipses with those already contained in the graph. If a duplicate vertex would be generated, then a loop closure occurs and the graph adjacency is updated. Further, all entries in the stack associated with the loop closure are moved to the top to prevent early backtracking.

This exploration method can be run online using local

Algorithm 2 Monitor Target Assignment

```
1: Input: Monitor Positions $p_m(t)$, Graph $\mathcal{G}$
2: Compute vertex position for each monitor: $\mathcal{V}(p_m(t))$
3: Generate virtual vertex $v_s$, with excess flow of $|\mathcal{T}_d|$
4: Add edges from $v_s$ to each $\mathcal{V}(p_m(t))$, with $c_{ij} = 0$
5: Generate virtual vertex $v_\tau$, with excess flow of $-|\mathcal{T}_d|$
6: Add edges from each $\mathcal{T}_d$ to $v_\tau$, with $c_{ij} = 0, \mu_{ij} = 1$
7: Compute min cost max flow from $v_s$ to $v_\tau$ through $\mathcal{G}$
   using Successive Shortest Paths            ▷ [39]
8: Follow edges with flow in reverse to return assignments
```

knowledge of obstacles. Further, by limiting graph expansion to flagged points on the frontier of exploration, we significantly reduce the number of vertices required to achieve the same levels of environmental coverage as sampling based methods. See our validation in Section VI-A for a comparison against a uniform sampling approach.

V. Monitoring of Discovered Targets

Section IV discussed the algorithms that enable our exploring agents to satisfy their subtask and generate a graph for the team to plan on. Here, we describe how our monitoring agents use this graph to both assign all discovered targets to nearby monitors and generate TSP tours that visit each one.

A. Target Assignment

As exploring agents discover tasks, we must assign them to the team of monitoring agents. Given the graph construction generated by ellipsoidal decomposition, $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{T}_d)$, we formulate the target assignment as a minimum cost maximum flow problem [39], [40]. We can write this problem as:

$$\min_{x_{ij}} \sum_{(i,j)\in\mathcal{E}} c_{ij} x_{ij}, \tag{6}$$

$$s.t \sum_{j:(i,j)\in\mathcal{E}} x_{(i,j)} - \sum_{k:(k,i)\in\mathcal{E}} x_{(k,i)} = 0 \, \forall \, i \neq s, \tau, \tag{7}$$

$$\sum_{(s,i)\in\mathcal{E}} x_{(s,i)} = \sum_{(i,\tau)\in\mathcal{E}} x_{(i,\tau)} = |\mathcal{T}_d|, \tag{8}$$

where c_{ij} is the cost incurred by travelling along the edge from vertex i to j and x_{ij} is the amount of flow travelling along the same edge. Constraint (7) ensures conservation of flow throughout the graph and (8) constrains the problem to ensure that each target is assigned once. This is possible by introducing two new vertices to the graph: a source, v_s, connected to each monitor position with $c_{si} = 0$, and a sink, v_τ, connected to each target with $c_{i\tau} = 0$ and maximum flow capacity of one. An example of this graph is depicted in Figure 2b.

To solve the linear program, we use the successive shortest paths algorithm to compute flow through the network from source to sink. Following this flow in reverse returns an optimal assignment of targets to monitoring agents according to edge weights, $c_{ij} \in \mathcal{E}$, and guarantees that each target is assigned to only one monitor. This process is described by Algorithm 2.

Algorithm 3 Monitor Tour Approximation

```
1: Input: Monitor Positions $p_m(t)$, Graph $\mathcal{G}$, Assigned targets $\mathcal{T}_m$
2: for $|\mathcal{T}_m|$ do
3:     Run Dijkstra from $t_m$                                   ▷ [41]
4:     Store shortest paths from $t_m$ to other targets $\mathcal{T}_m$
5:     Store shortest path from $t_m$ to $p_m(t)$
6: Generate a temporary fully connected graph with $\mathcal{V} = \mathcal{T}_m \cup p_m(t)$ and edge weights from the stored shortest paths' distances
7: Run Christofides' Algorithm on temporary graph               ▷ [42]
8: Move $p_m(t)$ towards next $t_m$ by following the stored path from Dijkstra
```

B. Target Touring

Once each monitor has received an assignment, defined as $\mathcal{T}_m$, they must plan a trajectory that visits each target. This can be accomplished by solving a travelling salesperson problem between targets. To generate tours that exclude all unnecessary vertices, we run Dijkstra's algorithm from each $\mathcal{T}_m$. We use the shortest distances between each $\mathcal{T}_m$ and $p_m(t)$ as edge weights in a temporary, fully connected graph, $\mathcal{G}_{temp}(\mathcal{V}_{temp} = [\mathcal{T}_m \cup p_m(t)], c_{temp})$, that only contains vertices for the monitor and assigned target positions. An example of this temporary graph is depicted in Figure 2c. In this setting, we can write the problem as:

$$\min_{x_{ij}} \sum_{i=1} \sum_{j \neq i} c_{ij} x_{ij}, \tag{9}$$

$$s.t \;\; x_{ij} \in 0, 1, \tag{10}$$

$$\sum_{i \neq j} x_{(i,j)} = 1, \tag{11}$$

$$\sum_{j \neq i} x_{(i,j)} = 1, \tag{12}$$

$$u_i - u_j + 1 \leq |\mathcal{V}_{temp}|(1 - x_{ij}) \tag{13}$$

$$\forall i, j : (1 \leq i \neq j \leq |\mathcal{V}_{temp}|),$$

where c_{ij} and x_{ij} are defined as the cost and flow traveling along edge (i, j) in $\mathcal{G}_{temp}$ and u_i is the number of edges travelled before reaching vertex i. Optimally solving the integer linear program is well known to be NP-Hard and, in highly complex environments with many targets, potentially intractable to be performed online. We instead choose to approximate these tours using Christofides' algorithm. Since this algorithm is of $O(\mathcal{V}_{temp}^3)$ complexity, we maintain polynomial complexity for all monitoring computations, which greatly improves efficiency in conjunction with the sparse graphs generated by the explorers. Further, if x^* is the true optimal flow in 9, then:

$$\sum_{i=1} \sum_{j \neq i} c_{ij} \tilde{x}_{ij} \leq \frac{3}{2} \sum_{i=1} \sum_{j \neq i} c_{ij} x^*_{ij}, \tag{14}$$

where $\tilde{x}$ is the flow returned by Christofides' algorithm [42]. To generate the tours, Christofides' algorithm first computes a minimum spanning tree (MST) in the graph, then finds a perfect matching for the subgraph of odd degree vertices in the MST. Joining the perfect matching and MST, an Euler tour is generated by visiting each node sequentially, twice. Finally, shortcutting is used to remove additional edges and prevent the tour from visiting the same vertex twice, returning an approximate TSP solution with bounded performance as in 14. Once a feasible TSP solution is computed, the path to the first node in the tour can be found using the results from the previously run Dijkstra's algorithm. Finally, if a monitoring agent has no assigned targets, we choose to move the agent towards the nearest discovered target. This allows idle agents to move towards more active regions of the environment and assist other monitors in servicing targets. This process of computing and following TSP tours on $\mathcal{G}_{temp}$ is outlined in Algorithm 3.

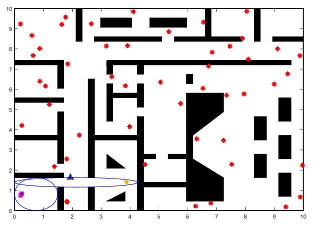
VI. Results

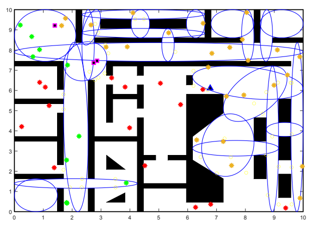
MATLAB simulations were performed to demonstrate the efficacy of our proposed team structure. These were performed in an environment meant to mimic the interior of a large building, with targets spread throughout its rooms and hallways. After demonstrating the performance of our approach, we also provide an analysis on the overall computational complexity of our algorithms.

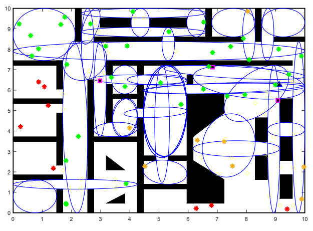
A. Simulations

We generated a non-convex environment containing 40 rectangular and triangular obstacles, depicted in Figure 3, to investigate our method's performance in a setting that could reflect indoor or cluttered environments. The team comprised one exploring agent and three monitoring agents, though the method trivially extends to multiple explorers by maintaining individual stacks for each agent. In each simulation, agents are deployed around a random initial position.

Coverage and Exploration Performance: To benchmark our performance, we compare our exploration algorithm to a uniform sampling method for generating the ellipsoidal graph, over 100 trials with random initializations. As a baseline, we generate an ellipsoidal graph with seed points from uniform sampling, and compare the overall free-space coverage against the graph resulting from our exploration algorithm (Algorithm 1). Figure 4 depicts the mean and variance of coverage, against the total number of ellipses needed to provide this coverage. As shown in the Figure, both methods reach similar final environmental coverage, however, our method requires significantly fewer graph vertices (ellipsoids). While uniform sampling requires full knowledge of the environment, our online exploration algorithm grows as the agent explores and discovers obstacles. While the ellipsoidal graph only reaches 86% coverage, we find the convex hulls that generate the ellipses cover up to 99.6% of the environment. The coverage of these convex hulls more accurately represents how much of the environment is reachable from the ellipsoidal graph. We also note that, without limitations on communication or implementation, it is possible to construct the graph from the convex hulls directly to reach near complete coverage of the environment within the graph.


(a) Timestep: 10


(b) Timestep: 150


(c) Timestep: 350

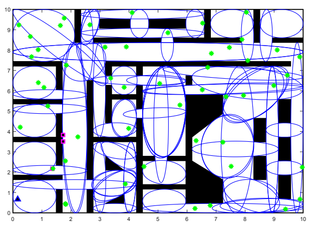

(d) Final Configuration

Fig. 3: Time series of simulated exploration and monitoring algorithms in unknown, non-convex environment

Monitoring Performance: Randomized trials were conducted to test the ability of our monitoring team to visit targets discovered by the exploration team (Algorithms 2 and 3). Across 100 simulations, each containing 50 randomly generated targets, monitors were able to service 98.4% of targets. This results in a total of only 79 out of the generated 5000 targets being missed. These results are expected as polytopic coverage, and therefore the method by which targets are discovered, is not 100%, however, this performance is satisfactory given the randomization and lack of environmental knowledge given to our team. Further, we did not control target generation to be accessible, nor did exploring agents know the true number of targets in the environment. In the future, we aim to expand our exploration algorithm to account for all edge cases and provide complete environmental exploration, which would result in a 100% target service rate.

B. Complexity Analysis

For the majority of planning, explorers are travelling between waypoints. The worst case complexity for this operation is travelling outside the current vertex, using Dijkstra's algorithm. This is well known to have a complexity of $O(\mathcal{E} + \mathcal{V}log(\mathcal{V}))$ with efficient implementation. If explorers are currently at a waypoint, then the algorithm runs the IRIS expansion incurring a complexity of $O(f(\eta)\mathcal{O})$, where $f(\eta)$ is a nonlinear term that is determined by the local geometry and user specified tolerance in the IRIS algorithm. After IRIS, each operation is of $O(1)$ aside from intersection calculation, which has complexity of $O(\mathcal{V})$. This means our exploration algorithm has two primary modes of complexity:

$$O(\mathcal{E} + \mathcal{V}log(\mathcal{V})) \quad \text{OR} \quad O(f(\eta)\mathcal{O}). \tag{15}$$

Though not strictly polynomial, due to the IRIS algorithm, one can enforce polynomial complexity by fixing the number of main iterations on IRIS with small performance penalties. In practice, due to the efficiency of IRIS and infrequent use of the algorithm, this is not necessary.

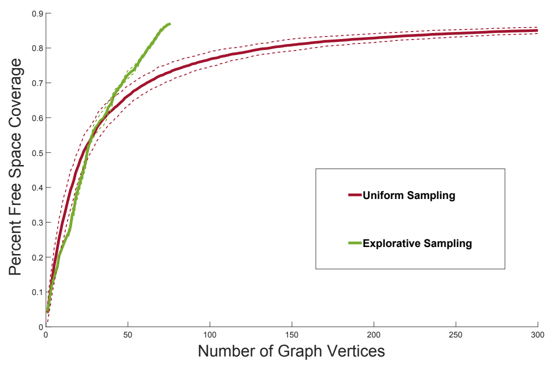


Fig. 4: Comparison of free-space coverage by our algorithm to uniformly sampled IRIS. Our approach achieves higher coverage with fewer ellipses.

While our exploration algorithm does not have strict polynomial complexity, our monitoring algorithms do. Starting with Algorithm 2, computing the vertex position of a monitor has complexity $O(\mathcal{V})$. Steps 3-6 are of $O(1)$. Finally, Successive Shortest Paths is well known to have a complexity of $O(U\mathcal{V}(\mathcal{E} + \mathcal{V}log(\mathcal{V})))$, where U is related to the maximum capacity of flow in the graph. While not inherently polynomial, the assignment problem is a special case where it can be shown that $U = 1$. Therefore the complexity is polynomial in: $O(\mathcal{V}(\mathcal{E} + \mathcal{V}log(\mathcal{V})))$ [39]. While optimally solving the travelling salesman problem is NP-hard, our implementation generates approximately optimal tours in strict polynomial complexity. Following Algorithm 3, steps 2-5 require running Dijkstra's algorithm from each assigned target for a complexity of $O(\mathcal{T}_m(\mathcal{E} + \mathcal{V}log(\mathcal{V})))$. Generating the temporary graph is $O(1)$ and contains one vertex for each assigned target and the monitor's current position. This results in Christofides' algorithm having complexity of $O(\mathcal{T}_m^3)$ [42]. Therefore, the complexity of generating tours is given by $O(\mathcal{T}_m(\mathcal{E}+\mathcal{V}log(\mathcal{V}))+\mathcal{T}_m^3)$. This complexity is then dominated by either target assignment or tour generation, depending on the size of the graph compared to the number of assigned targets. Then the overall monitoring complexity is given by:

$$O(\mathcal{V}(\mathcal{E} + \mathcal{V}log(\mathcal{V})) + \mathcal{T}_m(\mathcal{E} + \mathcal{V}log(\mathcal{V})) + \mathcal{T}_m^3), \tag{16}$$

which is strictly polynomial. This, in conjunction with the reduced graph sizes generated by our exploration algorithm, results in very fast execution of high level team planning, which is crucial in some applications.

VII. Conclusions

In this paper, we proposed a heterogeneous teaming solution to the problem of servicing targets in unknown, non-convex environments. Dividing our team into explorers and monitors based on their capabilities, we have demonstrated that our method efficiently explores and maps these environments into a lightweight, safe graph decomposition that allows the team to swiftly execute high level planning to accomplish the objective. This method has many potential applications, from firefighting to delivery services, where quick planning and safe execution are critical.

References

[1] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *The International journal of robotics research*, vol. 23, no. 9, pp. 939–954, 2004.

[2] P. Stone and M. Veloso, *Artificial Intelligence*, vol. 110, no. 2, pp. 241–273, 1999.

[3] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation," *The International journal of robotics research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[4] M. Mishra, P. Poddar, R. Agrawal, J. Chen, P. Tokekar, and P. B. Sujit, "Multi-agent deep reinforcement learning for persistent monitoring with sensing, communication, and localization constraints," *IEEE Transactions on Automation Science and Engineering*, pp. 1–0, 2024.

[5] Y. Ding, B. Xin, L. Dou, J. Chen, and B. M. Chen, "A memetic algorithm for curvature-constrained path planning of messenger uav in air-ground coordination," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 4, pp. 3735–3749, 2022.

[6] J. Parker, E. Nunes, J. Godoy, and M. Gini, "Exploiting spatial locality and heterogeneity of agents for search and rescue teamwork," *Journal of Field Robotics*, vol. 33, no. 7, pp. 877–900, 2016.

[7] M. W. Ulmer and B. W. Thomas, "Same-day delivery with heterogeneous fleets of drones and vehicles," *Networks*, vol. 72, no. 4, pp. 475–505, 2018.

[8] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[9] M. Schyns, "An ant colony system for responsive dynamic vehicle routing," *European Journal of Operational Research*, vol. 245, no. 3, pp. 704–718, 2015.

[10] A. Prorok, M. Ani Hsieh, and V. Kumar, "Fast redistribution of a swarm of heterogeneous robots," *EAI Endorsed Transactions on Scalable Information Systems*, vol. 3, no. 10, pp. 249–255, 12 2016.

[11] K. Yu, J. M. O'Kane, and P. Tokekar, "Coverage of an environment using energy-constrained unmanned aerial vehicles," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 3259–3265.

[12] M. Santos, Y. Diaz-Mercado, and M. Egerstedt, "Coverage control for multirobot teams with heterogeneous sensing capabilities," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 919–925, 2018.

[13] M. Coffey and A. Pierson, "Covering dynamic demand with multi-resource heterogeneous teams," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 11 127–11 134.

[14] M. S. Mondal, S. Ramasamy, J. D. Humann, J.-P. F. Reddinger, J. M. Dotterweich, M. A. Childers, and P. Bhounsule, "Optimizing fuel-constrained uav-ugv routes for large scale coverage: Bilevel planning in heterogeneous multi-agent systems," in *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2023, pp. 114–120.

[15] W. Babincsak, A. Aswale, and C. Pinciroli, "Ant colony optimization for heterogeneous coalition formation and scheduling with multi-skilled robots," in *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2023, pp. 121–127.

[16] M. Coffey and A. Pierson, "Assessing reputation to improve team performance in heterogeneous multi-robot coverage," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 2571–2577.

[17] E. Jones, B. Browning, M. Dias, B. Argall, M. Veloso, and A. Stentz, "Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 570–575.

[18] N. Wilde and J. Alonso-Mora, "Designing heterogeneous robot fleets for task allocation and sequencing," in *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2023, pp. 156–162.

[19] N. Basilico, T. H. Chung, and S. Carpin, "Distributed online patrolling with multi-agent teams of sentinels and searchers," in *Distributed Autonomous Robotic Systems*, N.-Y. Chong and Y.-J. Cho, Eds. Tokyo: Springer Japan, 2016, pp. 3–16.

[20] S. Hood, K. Benson, P. Hamod, D. Madison, J. M. O'Kane, and I. Rekleitis, "Bird's eye view: Cooperative exploration by ugv and uav," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*, 2017, pp. 247–255.

[21] C. Papachristos, S. Khattak, F. Mascarich, and K. Alexis, "Autonomous navigation and mapping in underground mines using aerial robots," in *2019 IEEE Aerospace Conference*, 2019, pp. 1–8.

[22] K. Leahy, Z. Serlin, C.-I. Vasile, A. Schoer, A. M. Jones, R. Tron, and C. Belta, "Scalable and robust algorithms for task-based coordination from high-level specifications (scratches)," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2516–2535, 2022.

[23] L. E. Beaver, R. Tron, and C. G. Cassandras, "A graph-based approach to generate energy-optimal robot trajectories in polygonal environments," *IFAC-PapersOnLine*, vol. 56, no. 2, pp. 7000–7005, 2023, 22nd IFAC World Congress.

[24] R. Chatila and J. Laumond, "Position referencing and consistent world modeling for mobile robots," in *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 138–145.

[25] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 12 2000.

[26] Y. Li, H. Chen, M. Joo Er, and X. Wang, "Coverage path planning for uavs based on enhanced exact cellular decomposition method," *Mechatronics*, vol. 21, no. 5, pp. 876–885, 2011, special Issue on Development of Autonomous Unmanned Aerial Vehicles.

[27] L. J. Guibas, J.-C. Latombe, S. M. Lavalle, D. Lin, and R. Motwani, "Visibility-based pursuit-evasion in a polygonal environment," in *Algorithms and Data Structures*, F. Dehne, A. Rau-Chaplin, J.-R. Sack, and R. Tamassia, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 17–30.

[28] P. T. Kyaw, A. Paing, T. T. Thu, R. E. Mohan, A. Vu Le, and P. Veerajagadheswar, "Coverage path planning for decomposition reconfigurable grid-maps using deep reinforcement learning based travelling salesman problem," *IEEE Access*, vol. 8, pp. 225 945–225 956, 2020.

[29] N. M. Stiffler and J. M. O'Kane, "Complete and optimal visibility-based pursuit-evasion," *The International Journal of Robotics Research*, vol. 36, no. 8, pp. 923–946, 2017.

[30] R. Deits and R. Tedrake, *Computing Large Convex Regions of Obstacle-Free Space Through Semidefinite Programming*. Springer International Publishing, 2015, pp. 109–124.

[31] A. Ray, A. Pierson, and D. Rus, "Free-space ellipsoid graphs for multi-agent target monitoring," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6860–6866.

[32] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[33] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the Society for Industrial and Applied Mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[34] M. Z. Spivey and W. B. Powell, "The dynamic assignment problem," *Transportation science*, vol. 38, no. 4, pp. 399–419, 2004.

[35] G. A. Mills-Tettey, A. Stentz, and M. B. Dias, "The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-07-27*, 7 2007.

[36] M. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[37] Y. Tang, Z. Ren, J. Li, and K. Sycara, "Solving multi-agent target assignment and path finding with a single constraint tree," in *2023 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, 2023, pp. 8–14.

[38] W. Hönig, S. Kiesel, A. Tinka, J. W. Durham, and N. Ayanian, "Conflict-based search with optimal task assignment," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, ser. AAMAS '18. International Foundation for Autonomous Agents and Multiagent Systems, 2018, p. 757–765.

[39] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows : theory, algorithms, and applications*. Englewood Cliffs, N.J: Prentice Hall, 1993.

[40] L. R. Ford and D. R. Fulkerson, "Maximal flow through a network," *Canadian Journal of Mathematics*, vol. 8, p. 399–404, 1956.

[41] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[42] N. Christofides, *Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem*, ser. Management sciences research report. Management Sciences Research Group, Graduate School of Industrial Administration, Carnegie-Mellon University, 1976.