Multi-Resolution Informative Path Planning for Small Teams of Robots

Nayana Suvarna

CMU-RI-TR-24-79

December 18, 2024

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Sebastian Scherer, *chair*
Jiaoyang Li
Ananya Rao

Submitted in partial fulfillment of the requirements for the degree of Masters of Science in Robotics.

To my village of family, friends, and mentors

Abstract

Unmanned aerial vehicles can increase the efficiency of information gathering applications. A key challenge is balancing the search across multiple locations of varying importance while determining the best sensing altitude, given each agent's finite operation time. In this work, we present a multi-resolution informative path planning approach for small teams of unmanned aerial vehicles. We model our problem as a team orienteering problem, aiming to maximize reward by performing searches over a set of spatially separated regions. We convert each region into a set of nodes across multiple fixed altitudes, and compute a cost and reward for each node based on sensing resolution at discrete altitudes. We utilize a linearization method to precisely capture the nonlinear information gain reward for each node, which allows us to leverage mixed-integer linear programming optimizers to solve our problem. Through this approach, we're able to generate plans for our team of agents that balance revisiting regions of importance and exploring new regions. We evaluate our approach against greedy, naive greedy, and random baselines for teams of up to three agents on multiple maps with varying information distributions. We show that our approach can produce plans of greater optimality within a fixed time limit and limited sensing budget over the baselines. We also discuss the tradeoffs in solution quality and runtime over the optimization process compared to the baseline solutions.

Acknowledgments

I'd like to first thank my advisor, Basti, for his mentorship and guidance over the course of my masters. When I started the program, I knew very little about drones and unmanned aerial vehicles. Through my experiences in the lab and through our various research conversations, I was able to gain an understanding of the real-world impact and challenges behind these systems. This knowledge served as the foundation for my thesis.

I'd also like to thank my committee member, Prof. Jiaoyang Li, whose extensive knowledge and expertise on multi-agent systems was instrumental in shaping the direction of this thesis. Thank you as well to my thesis member Ananya Rao for her friendship and mentorship over the years and for her guidance on my thesis document and slides.

I would like to extend a large amount of gratitude to my main collaborators for this work. I'd first like to thank Brady Moon, for being an exceptional friend, mentor, and desk neighbor. Working alongside you (both literally and figuratively) across our various informative path planning projects has been one of my greatest pleasures. Next, I'd like to thank Charles Noren for his knowledge of operations research that helped influence this work. Thank you both for being my sounding boards and for our countless research conversations from the initial ideas of this thesis to the finest details of the final document. I'm truly grateful for the opportunity to have worked with the two of you. Thank you as well to many members of the AirLab who helped contribute to its culture and made working in the lab an enjoyable experience - Jay P., Cherie, Jay K., Ian, and Mateo.

This thesis is the culmination of my six-year journey in the RI and it would not have been possible without the large community of people that have supported me along the way. This is partially thanks to the people who helped shape my robotics experience as an undergrad. Thank you to Prof. Howie Choset with whom I worked with for the majority of my undergrad. The practical hands-on experience I gained in his lab taught me how to effectively communicate my research ideas and served as the foundation for my robotics knowledge. Thank you as well to my undergrad advisor, Dr. Dickerson, for always believing in me and for pushing me to always do my best. Your support for me during the IAC helped show me that I could achieve anything I set my mind to.

I'd also like to thank Prof. John Dolan and Rachel Burcin for their support

and mentorship through my time in the Robotics Institute Summer Scholars (RISS) program for two summers. Through these experiences, I was able to build my confidence as a researcher. Your mentorship and guidance over the years have been vital to my personal and professional growth. I would not be in this program if it weren't for the two of you urging me to apply.

I'd like to thank my friends in my MSR cohort - Sofia, Sam, Conner, and Matt for helping make long days in the office more bearable with our lunch breaks and little treat runs. I would also like to thank the many friends that I've met through my time at CMU - Abby, Ananya, Bart, Ben E., Ben F., Dan, Min, Mrinal, Ravi, Rayna, Swapnil, Tejus, Vivian, Alex, Sam, Maggie, and Pranav. Through our countless coffee shop work sessions, walks around SQH, Blood on the Clocktower games, and various other hangouts, thank you for helping me build a community at CMU and for making me feel welcome in the RI.

Having lived in Pittsburgh for the past seven years, there are countless friends outside of CMU who have supported and helped shape me into the person I am today. To my oldest friends - Steph and Josh, you both know me better than I know myself and I don't know who I'd be without you. To my other Pittsburgh friends - Tom, Prem, and Ashumi, thank you for making Pittsburgh feel like home. Thanks as well to my friends from the IAC - Andrew and Cindy for being invaluable mentors and friends and for letting me take advantage of your wealth of wisdom and advice.

Finally, thank you to my family for being my biggest cheerleaders. My experiences and accomplishments are a direct reflection of the sacrifices, love, and encouragement that you have continuously given me. Your belief in me has been a constant source of motivation, and I could not have come this far without this unwavering support.

Funding

This thesis was support by ONR grant N000142212548 and contract N6833522C0179.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Autonomous robots are used for a variety of information gathering applications including environmental monitoring [6] [11], inspection [28], and disaster response [3]. These robots can be leveraged to tackle tasks that may be dangerous or even infeasible for humans to complete. In these situations, a prior is often defined that represents the distribution of information in the search space. This priori information distribution could represent probable locations for survivors in a search and rescue situation or areas of wear and tear for inspection. Using this prior, robots can be guided toward areas of higher interest in their search. Teams of robots can be used to improve the speed and efficiency of these information gathering tasks.

We explore the multi-agent information gathering problem where a team of unmanned aerial vehicles seek to plan informative paths over a large search space to search for objects of interest. We consider a scenario with a maximum team size of three robots where the collective flying budget allocated to the team of agents is smaller than the budget needed to cover the entire space. Thus, agents must balance the trade-off between exploring new areas of interest and exploiting existing information to revisit areas of higher interest at varying sensing resolutions.

1.2 Challenges

The main challenge at the core of information gathering problems is that each robot has a finite operating time. This can be quantified through flight time, travel distance, or battery levels. For this work, we represent the limited sensing resources of the robot as flight distance or budget measured in meters. This challenge is exacerbated by having non-uniform expected information over the search space as agents have to balance observing multiple areas with varying importance. Agents have to use the information to plan paths that maximize their overall team reward, modeled as information gain, while not exceeding their individual flight budgets. This is often formulated as the informative path planning problem.

Planning for multiple robots introduces additional challenges because the search space grows exponentially as the number of agents increases [25]. This may make it more difficult or potentially even infeasible to produce optimal plans. During the planning procedure, multiple potentially competing objectives across the set of agents must be balanced to produce plans that maximize overall team performance.

We introduce additional complexities to the informative path planning problem by allowing agents to visit areas of interest multiple times at varying fixed-altitude sensing resolutions. Flying at a lower altitude results in a smaller field of view, which enables a higher sensing resolution. This limited field of view, however, can lead to a large amount of budget to be expended when searching over a large space. At higher altitudes, the field of view is larger, which allows for less budget to be expended during a search. However, the sensing resolution is lower, which results in poorer sensing performance. Thus, for individual sensing measurements, there is a trade-off between sensing quality and budget expended. Compounding this across multiple measurements at varying sensing resolutions, the cost versus benefit trade-off of multiple observations taken over a single region must be considered as part of our problem formulation.

1.3 Contributions

In this work, we present a multi-agent informative path planning approach modeled as a team orienteering problem to plan informative paths for a small team of up

to three unmanned aerial vehicles. We formulate it as a task allocation framework where tasks are represented as searches over a set of regions of varying importance at multiple fixed altitudes. We compute a cost and reward for each task, which we represent as nodes, based on the sensing resolution at its fixed altitude. By leveraging a linearized version of our information gain function, we're able to utilize existing mixed integer linear programming solvers for our problem.

For our results, we compare our approach against naive, greedy, and random baselines on multiple maps with varying information distributions. We demonstrate that our approach is able to consistently outperform our baselines at varying limited budgets. In our discussion, we provide an analysis of the runtime versus performance improvement achieved when utilizing our approach. We also observe the differences in performance for our system across varying team sizes.

Chapter 2

Background

2.1 Informative Path Planning

The informative path planning problem (IPP) consists of planning a set of paths that maximize information gain, subject to a robot's budget constraints. It is shown to be NP-hard. Many approaches have been developed to compute optimal or close to optimal solutions for this problem.

Graph-based approaches abstract the environment as a set of nodes and edges, where the nodes consist of potential sensing locations. Some works utilize branch and bound to solve the problem [4] or present greedy approaches to approximate solutions [23] [5]. Other approaches formulate the graph search problem as an orienteering problem [1], [26]. Although these approaches can generate high-quality solutions, they become difficult to solve as the number of nodes in the graph increases. This can lead to poor solutions in large, dynamic, and high-dimensional search spaces.

For large, dynamic environments, existing works have explored utilizing sampling-based planners. A tree is built by sampling states in the search space. Then, the path in the tree with the highest information gain is returned at the end of the allocated planning time. Some works randomly sample states to build the tree [12], while others bias sampling using the prior fed in [16]. Other approaches [21] rewire the tree generated to reduce the tree size while simultaneously improving the quality of existing plans. These approaches handle large search spaces more effectively, but they can still struggle to efficiently sample states and generate plans when the

dimensionality of the search space increases. Additionally, they are unable to provide a measure on the quality of plans generated with respect to the true optimal solution to the problem.

2.2 Multi-Agent Information Gathering

Many approaches have been developed to tackle the multi-agent information gathering problem. One way of formulating the problem has been through region partitioning where each agent gets a single region to search [13], [9], [22]. These approaches fail to capture the nuances introduced by performing a search over a non-uniform prior.

A common way of formulating the information gathering problem over a non-uniform prior is through a task allocation problem. With these approaches, the tasks consist of a set of sensing locations to visit. Each sensing location has a reward that ties to its associated amount of information. So, the objective acts similarly to the single agent IPP problem except now the team of multiple agents has to maximize their collective reward.

One way of approaching the task allocation problem is through optimization-based approaches. For these approaches, an exact solver such as a mixed-integer linear programming solver [8] could be used to plan paths for the set of agents. A key downside, however, to these approaches is that the exact solvers used for these approaches can take a long time to solve to optimality. This can lead to difficulties in producing plans when searching over a large space or when trying to plan for a large team of agents. So, existing approaches rely on metaheuristics such as ant-colony optimization [7], or genetic algorithms [14] to approximate sub-optimal solutions with the tradeoff of faster solving times.

2.3 Multi-Resolution Information Gathering

Outside of these approaches, the problem of multi-resolution planning for unmanned aerial vehicles has been looked at by other groups. In [19], the authors present an evolutionary algorithm for environmental mapping. In [20], the authors explore a multi-resolution coverage-based method. In other works, [1], [2], the authors present

approaches that plan across multiple altitudes when searching for objects of interest. The multi-agent problem with multi-resolution sensing for unmanned aerial vehicles has largely been unexplored in existing literature. Especially when coupled with information gain as part of the objective function. Outside of the multi-agent aspect of our work, we also explore how geometric regions, representing clusters of sensing locations, can be used as the tasks allocated to the team of agents instead of sensing locations alone.

Chapter 3

Multi-Agent Multi-Resolution Informative Path Planning

3.1 Problem Definition

Let $\mathcal{T} = \{T_0, T_1, \ldots, T_n\}$ represent trajectories for a small team of agents. $C(\mathcal{T}_i)$ and $I(\mathcal{T}_i)$ represent the respective cost and reward or information gain of trajectory T for agent i. We then define the multi-agent informative path planning problem as follows where $\mathcal{T}^*$ is the set of trajectories from the total set of trajectories $\mathbb{T}$ for a set of agents that maximizes information gain without exceeding the budget constraint B_i for for each agent i.

$$\mathcal{T}^* = \underset{\mathcal{T} \in \mathbb{T}}{\operatorname{argmax}} \sum_{i=1}^{n} I(\mathcal{T}_i) \text{ s.t. } C(\mathcal{T}_i) \leq B_i \tag{3.1}$$

To generate these paths, we model the multi-agent informative path planning problem as a team orienteering problem to jointly optimize trajectories for all agents. In this chapter, we break down the various components of our system that feed into this formulation. A summary of our approach can be seen in Fig. 3.1.

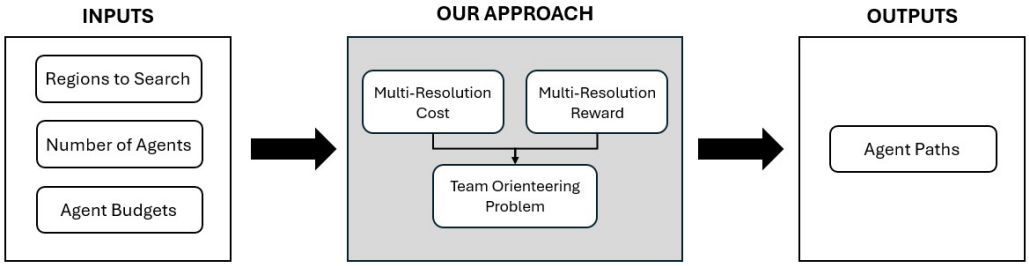

Figure 3.1: Diagram outlining the inputs and outputs of our approach and the individual components of our system. The inputs consist of the regions to search and information on the set of agents to plan for. The outputs include a set of paths for all agents.

3.2 Region-Based Map Representation

We represent our environment as a set of spatially separated regions consisting of areas and line segments where each region is an abstract representation of a cluster of sensing locations. Areas are represented as a set of vertices that form a closed polygon. They can be used to represent environmental features such as fields, forests, and city blocks. Line segments are represented as a set of two points that form a line. They can be used to represent environmental features such as trails, rivers, and roads. Each area and line segment is assigned a uniform probability $P(X)$ where X represents the likelihood that objects of interest lie within the region.

To convert this geometric representation into a graph, we abstract each region as a node. We begin by computing the centroid of each region. This gives us a two-dimensional representation of all our regions. We then place these at various fixed altitudes that the team of agents fly at. For our approach, we assume that the agents are flying at a maximum of two fixed altitudes. Thus, each node can be summarized as (x, y, z, c, r) where x and y are the centroids for the region and z corresponds to a fixed altitude that a search of the region is performed at. c and r then give us the respective budget spent and information gain received after completing the search over the region.

We assume that we have a set of homogenous agents that can perform a search at any altitude. So, our graph is fully connected. However, this underlying graph representation can be modified to reflect problem-specific constraints such as obstacles

or restrictions on the altitudes that specific agents can fly at. During this process, we also generate a mapping that represents the lower altitude searches that are encompassed by searches at the higher altitude.

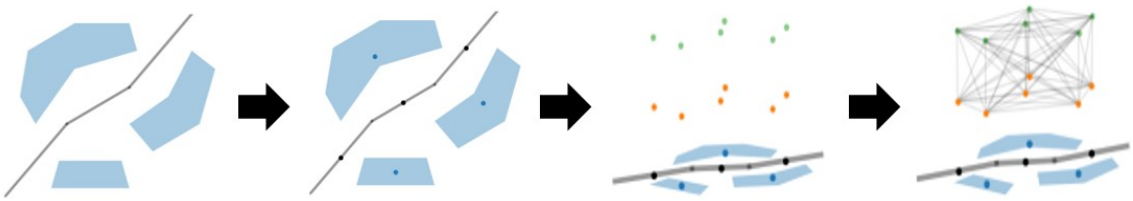

Figure 3.2: Visualization of the process used to generate a graph representation for a set of geometric regions. We first compute the centroids for each region. Then, we place these centroids at varying altitudes to form our set of nodes. Finally, we connect all these nodes to form a fully connected graph

3.3 Multi-Resolution Costs

For each node, we define a cost that represents the flying budget in meters needed to perform a search over the region at the desired altitude. We employ two different methods for calculating cost depending on whether the region is an area or a line segment.

For areas, we compute a coarse grid representation within the bounding box the polygon is inscribed in where the dimensions of each cell are equivalent to the sensor footprint of an agent at the node altitude. We assume that each agent has a downward-facing camera so that this footprint can be represented as a square based on the sensor configuration. We filter these cells into a set of nodes where our set of nodes consists of the cells that intersect the polygon. We treat the upper leftmost node as our start node, and the lower rightmost node as our end node. We then compute a coverage path that passes through all nodes using a wavefront coverage planner [27]. We treat the Euclidean distance of this path as the cost estimate for a given area. By using the dimensions of the agent's sensor footprint as the cell size, we're able to adjust the budget of an area based on sensing resolution.

For line segments, we assume that to cover the region, a robot would travel the length of the line. So, we treat the Euclidean distance of the line segment as our base cost. We adjust this based on sensing resolution by subtracting the length of the

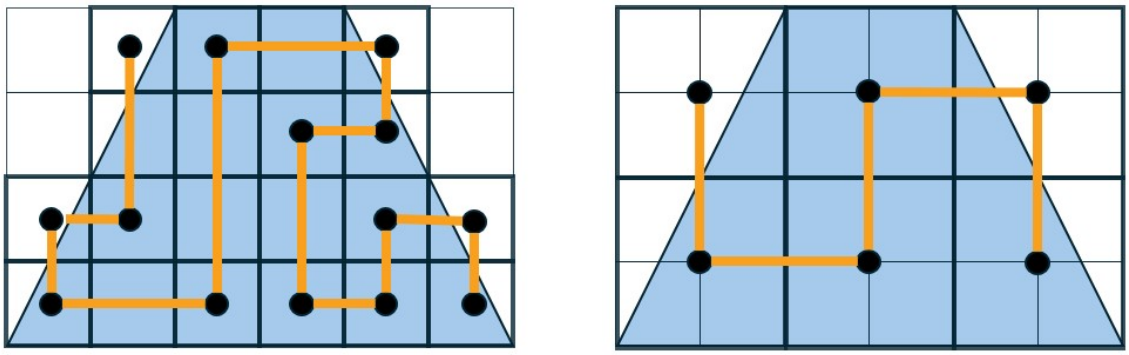

Figure 3.3: Visualization of the path generated at two different altitudes. In the left image, the sensor footprint is small to reflect the higher sensing resolution. In the right image, the sensor footprint is larger to reflect a low sensing resolution

sensor footprint from this base cost. Formulating it this way allows us to similarly adjust the budget of a line segment based on sensing resolution.

3.4 Information Reward Function

Our reward function for information depends on having an accurate model of the performance of the sensor onboard each vehicle. For our implementation, we consider an electro-optical range-based sensor, though this framework could extend to other sensors. Let Z represent a detection. Thus, we can model the performance of our perception system through its true positive rate $P(X|Z)$ and false positive rate $P(\neg X|Z)$. In the example sensor model in Fig. 3.4, we can see that as the range of observations increases, the sensor performance degrades until it plateaus to 0.5 where observations have minimal effect.

Using this sensor model, we compute our information reward function as the reduction in entropy [18] from a new measurement Z. We model entropy using Shannon entropy through

$$H(X) = -P(X)\log P(X) - P(\neg X)\log P(\neg X) \tag{3.2}$$

Through calculating the entropy of $P(X)$ and $P(X|Z)$, we can then calculate the reduction in entropy as

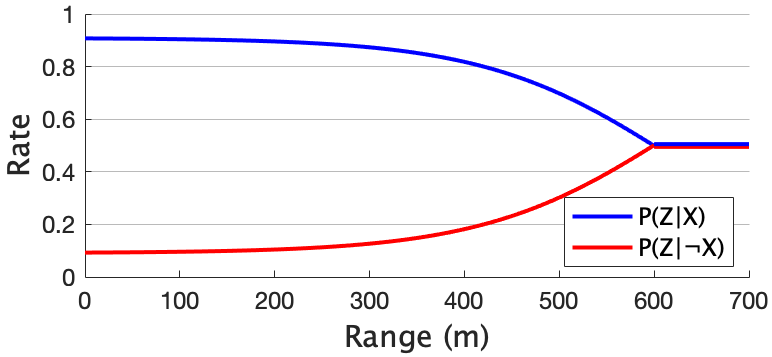

Figure 3.4: An example sensor model [16]. The blue line represents the true positive rate $P(X|Z)$ and the red line represents the false positive rate $P(\neg X|Z)$

$$\Delta H(X|Z) = H(X) - H(X|Z) \tag{3.3}$$

When applying multiple observations over X, the final belief and, consequently, $\Delta H(X|Z_1, Z_2)$ are the same regardless of the order of measurements. This is due to the commutative property of the Bayesian updates and Shannon entropy calculations. Thus, we can pre-compute a reward given a single low or high observation or multiple observations. This intuition can be leveraged to compute a linear formulation of the non-linear $\Delta H(X|Z)$ function.

3.5 Multi-Resolution Rewards

For each node, we also define a reward that represents the information gained after completing a search over a region at the node's altitude. At a high level, we multiply the reward we precomputed for a high, low, and combined high and low observation over the set of cells that fall within a region. We vary the method we use to compute the number of intersecting cells based on whether a region is an area or a line segment.

For areas, we first compute a grid within the bounding box in which an area is inscribed. We then multiply the rewards we precomputed by the number of cells that intersect with the shape of the area. For line segments, we multiply the rewards we precomputed by the number of cells that intersect with the line through voxel traversal. Through utilizing these methods, we're able to vary the reward for a region

based on sensing resolution and the number of observations. Additionally, this allows us to pre-compute the reward over a given region across all combinations of high and low observations.

3.6 Team Orienteering Problem Formulation

Bringing all these components together, we model the multi-agent informative path planning problem as a Team Orienteering Problem. The Team Orienteering Problem is a multi-agent extension of the Orienteering Problem [24] where the objective function is a reward maximization problem subject to a set of resource constraints.

Let $I = \{1, 2, \ldots, P\}$ represent an index set for the P agents in the team. The agents in the team are tasked with performing a search over a set of regions at two fixed altitudes. Let $L = \{l_1, l_2, \ldots, l_{m_1}\}$ denote the set of nodes representing the search at the low altitude and let $H = \{h_1, h_2, \ldots, h_{m_2}\}$ denote the set of nodes representing the search at the higher altitude. All agents are required to begin their paths at start locations, $D^- = \{d_1^-, d_2^-, \ldots, d_P^-\}$, and end their paths at end locations, $D^+ = \{d_1^+, d_2^+, \ldots, d_P^+\}$.

All agents share a workspace that can be represented as a weighted undirected graph $G = (V, E)$. The vertex set $V = L \cup H \cup D^- \cup D^+$ represents the set of all possible locations for all agents and the edge set $E = \{(v_i, v_j) \in V \times V \mid i \neq j\}$ represents the connections between these vertices. Thus, the set of tasks the agents have to complete can be expressed as $S = V \setminus (D^+ \cup D^-)$.

Each vertex v_i has an associated tuple (X^i, Y^i, Z^i, C^i) where X^i and Y^i represent the centroid of the region, Z^i represents a measurement over the region at the fixed altitude, and C^i represents a constant probability for the likelihood of objects of interest lying within the region described by the node. Each vertex v_i has an associated reward $R_i = \Delta H(C^i|Z^i)$ that corresponds to the reward obtained after performing a search over the region associated with the node. Each edge (v_i, v_j) has a weight $(b_{ij} + b_j)$ where b_{ij} is the Euclidean distance from v_i to v_j and b_j represents the positive cost of performing the search task at v_j.

We define a function $f : L \to H$ that maps low and high-altitude nodes. Given a low-altitude node $l_i \in L$, the function f returns a corresponding high-altitude node $h_j \in H$. This mapping indicates the search at the lower altitude node that

is encompassed by the search at the higher altitude. In cases where multiple low altitude searches are encompassed by a high altitude search, h_j could map to multiple elements $l_i \in L$. Using this mapping, we compute a reward $R_{ij} = \Delta H(C^i|Z^i, Z^j)$ for viewing a region at both the higher and lower altitudes.

For the mixed-integer linear programming model, we define three different decision variables. The first is a binary decision variable that denotes whether the edge connecting (v_i, v_j) was traversed by agent $p \in I$.

$$x_{ijp} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E \text{ is traversed by agent } p \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

The next decision variable is a binary variable that denotes whether vertex $v_i \in V$ was visited by agent $p \in I$.

$$y_{ip} = \begin{cases} 1, & \text{if } v_i \text{ is visited by agent } p \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

The third decision variable is a binary variable we define through an indicator constraint that indicates whether two nodes $v_i, v_j \in V$ have been visited by any agent $p \in I$ in the team of agents.

$$m_{ij} = \begin{cases} 1, & \text{if } \sum_{p=1}^{P} y_{ip} + \sum_{p=1}^{P} y_{jp} \geq 2.0, \\ 0, & \text{if } \sum_{p=1}^{P} y_{ip} + \sum_{p=1}^{P} y_{jp} < 2.0 \end{cases} \tag{3.6}$$

The right-hand side of the first constraint is set to 2 so that m_{ij} is set to 1 when both v_i and v_j are visited. The right-hand side of the second constraint is set to 1.9 so that this value is set to 0 otherwise. For our implementation, we assume that the size of a sensor footprint is larger than the size of a region. So, there is a one-to-one mapping between high and low-altitude nodes. We can then define our objective function as

$$\max \sum_{p=1}^{P} \sum_{i \in L} R_i y_{ip} + \sum_{p=1}^{P} \sum_{j \in H} R_j y_{jp} + \sum_{k \in L} (R_{kf(k)} - R_k - R_{f(k)}) m_{kf(k)} \tag{3.7}$$

By formulating our objective function this way, we're able to obtain a linear representation of the dependent rewards from single and multiple views over a region. The first component of the summation computes the reward for all individual low-altitude nodes $i \in L$. The second component of the summation computes the reward for all individual high-altitude nodes $j \in H$. The total reward for both views is not equal to the summation of both individual rewards. So, we add a third component to ensure that the summed reward from the objective function is equivalent to the reward from both altitudes. The third component of the summation iterates through low altitude node $k \in L$ and the mapped high altitude node $f(k) \in H$ to compute the combined reward from both high and low views over a region. The $m_{kf(k)}$ term is 1 when both v_k and $v_{f(k)}$ are visited. The individual reward R_k for a low altitude view and the individual reward $R_{f(k)}$ for a high altitude view are subtracted from the combined reward $R_{kf(k)}$ to ensure that the individual rewards are not double counted from the previous summations. The constraints that we use to optimize this objective can be formulated as

$$\sum_{p=1}^{P} \sum_{i \in T} x_{d_p^- ip} = \sum_{p=1}^{P} \sum_{j \in T} x_{jd_p^+ p} = P \tag{3.8}$$

$$\sum_{p=1}^{P} y_{kp} \leq 1 \quad \forall k \in T \tag{3.9}$$

$$\sum_{i \in T} \sum_{j \in T \setminus \{i\}} x_{ijp}(b_{ij} + b_j) \leq B_{\max}, \quad \forall p \in I \tag{3.10}$$

$$\sum_{i \in D^- \cup T} x_{ikp} = \sum_{j \in D^+ \cup T} x_{kjp} = y_{kp} \quad \forall k \in T; \forall p \in I \tag{3.11}$$

$$u_{ip} - u_{jp} + 1 \leq (|V| - 1)(1 - x_{ijp}) \quad \forall i, j \in V \setminus v_s \; \forall p \in I \tag{3.12}$$

$$2 \leq u_{ip} \leq |V| \quad \forall i \in V \setminus v_s \; \forall p \in I \tag{3.13}$$

$$x_{ijp}, y_{ip} \in \{0, 1\} \quad \forall i, j \in V; \forall p \in I \tag{3.14}$$

Constraint (3.8) ensures that all agents begin from a start node v_s and end at a desired goal node v_g. For our application, we treat v_g as a dummy node where the distance from a given node $v_j \in V$ to v_g is 0. Thus, agents can end their paths at any node in the graph. Constraint (3.9) ensures that the team of agents visits each node at most once. Constraint (3.10) ensures that the cost of each agent's trajectory does not exceed the agent's allocated budget. Constraint (3.11) ensures the connectivity of each agent's path. Constraint (3.13) defines a u variable that dictates the order of node v_i in the path of agent p. This variable is used in Constraint (3.12) through Miller-Tucker-Zemlin subtour constraints [15] to ensure that no subtours exist in the paths of the agents.

Chapter 4

Experimental Setup

4.1 Map Generation

Although our method is capable of handling both areas and line segments, we focus the evaluation of our method on maps with regions exclusively represented as areas. This is due to the large discrepancy in the reward and cost associated with line segments and areas. Including both region representations requires an application-dependent weight that balances the difference in reward-to-cost ratios to ensure that agents search both line segments and areas. Limiting our focus to areas allows us to make a more straightforward comparison between regions and gives us a clearer assessment of our method's performance. To evaluate our method, we rely on a city and wildlife environment from real-world locations.

Figure 4.1: Map generation process for the city environment

For the city environment, we consider a scenario where we are trying to find potential survivors after a natural disaster. We generate our map using data from

Boston, Massachusetts. We leverage the inherent structures that the various roadways in the city provide to generate the set of regions. We generated the city environment by filtering OpenStreetMap [17] data to include highways and major roads, buffering each based on width and merging these buffered roadways. Subtracting the union of all these buffered roads from the convex hull of our search area gives us the set of regions enclosed by roadways. Thus, the set of areas for our search consists of various city blocks within our search area. An overview of this map generation process can be seen in 4.1 and the city environment we use can be seen in Figure 4.2.

For the wildlife environment, we consider a scenario where we're trying to find a set of cows grazing on a mountain. The map for this search area comes from an area in the Utah mountains. To generate our set of regions, we place a set of areas around each of the lakes in the environment and open grassy areas to represent probable grazing locations for the cows. The wildlife maps we use for our experiments can be seen in Figure 4.2

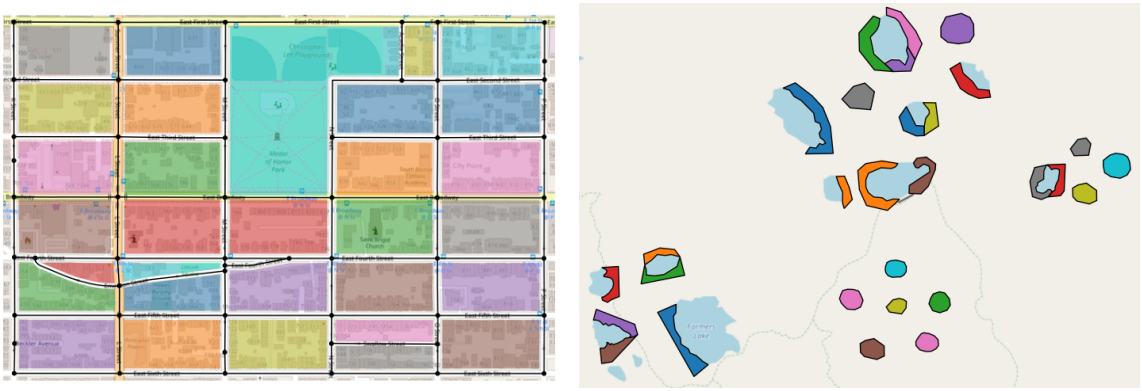

Figure 4.2: Visualizations of the two maps we used for our experiments

4.2 Baseline Methods

To evaluate our method, we implemented three different baselines: a random, naive greedy, and greedy algorithm. At a high level, we use a sequential planning method for each of our baseline methods. The main difference between our methods lies in the node selection process.

Pseudocode for our various baseline methods can be seen in Algorithm 1. The

Algorithm 1 Multi-Agent Sequential Planner

Input: $B = [b_0, b_1, \ldots, b_n]$,$\mathcal{V}$,$v_0$,$f$
Output: informative paths for all agents

$i \leftarrow 0$
$P \leftarrow |B|$
$rem\_budget = B[i]$
$curr\_node = v_0$
while $i < P$ **do**
    $F \leftarrow getFeasibleNodes(\mathcal{V}, rem\_budget, curr\_node)$
    if $F = \emptyset$ **then**
        $i \leftarrow i + 1$
        $rem\_budget \leftarrow B[i]$
        $curr\_node \leftarrow v_0$
    else
        $R \leftarrow getBestNode(F, curr\_node)$
        $nodes \leftarrow nodes \setminus \{R\}$
        $rem\_budget \leftarrow rem\_budget - euclid\_dist(curr_node, R)$
        $curr\_node = R$
    end if
end while

algorithm plans for the set of agents by planning for them sequentially until the agent index i is equivalent to the number of agents P. During each loop iteration, a node is selected for agent i to visit. We keep track of the current node *curr_node* and remaining budget *rem_budget* for agent i. We begin by filtering down the set of unvisited nodes in the set of nodes, V, to form a set of feasible nodes F for the agent to visit using the function *getFeasibleNodes*. We classify a node as feasible if the budget required to travel to a node from the current node an agent's at and the budget needed to perform the search task at a node is less than the remaining budget *rem_budget* for an agent. If the set of feasible nodes is empty, then we know that we've expended the budget of the agent and plan for the next agent by incrementing the agent index i by 1 and by resetting the current node to the start node and the remaining budget as the max budget for the next agent.

Using the set of feasible nodes, we select the best node by returning the node from the set of feasible nodes with the largest reward. Our method for computing reward differs based on the planner that we're using. For the random planner, we

assign a random reward for each node. For the naive greedy planner, we compute the reward for a node similar to our objective function using the number of views over a region. For the greedy reward, we divide this node reward by the cost of traveling to the node and completing a search task. Through these three reward functions, we observe the differences between randomly sampling nodes, exclusively considering reward, and balancing reward and cost for the agent trajectories.

The node with the best reward is returned and removed from the set of nodes to mark that it's been visited by agent i. This is similar to the behavior from Constraint 3.9 to ensure that each node is only visited once. We also mark the node as visited so it can be used for reward calculations. Finally, this node is set as the current node of the agent so it can resume planning from this position during the next planning loop.

4.3 Experimental Method

We evaluate our approach against the baselines in 10 tests per budget for each map type with randomized information maps for each run. To generate each information map, we assign a random probability in the range of 0.0 to 0.5 to each region, to represent the likelihood that objects of interest lie within the region. To determine the budget allocated to each agent, we calculate a coverage budget estimate for the map. We compute this coverage budget as the summation of the cost to perform the search at each node in the map plus an estimate of the cost to transition between every node. We calculate this transition cost by running a greedy traveling salesman problem on all the nodes in our map. To determine the set of budgets for our tests, we increase this budget in 20% increments and divide each increment by the number of agents to determine the budget allocated to each agent.

To compare our method against the baselines, we compare the suboptimality of our solution after an early stop to the optimal solution. We define the optimal solution for this comparison as the solution found when the optimizer reaches a 1% gap or the solution found after an 1800 second maximum planning time. We use a 120 second early stop as the solution for our approach. Through this metric, we are able to measure the increase in optimality that our approach provides over the baselines when faced with a finite planning time and limited budget.

For the city environment, the agents fly at a fixed altitude of 80 meters and 100

meters with a true positive rate and true negative rate of 0.75 for the lower altitude and 0.70 for the higher altitude. For the wildlife environment, we fly at a fixed altitude of 120 meters and 150 meters with a true positive rate and true negative rate of 0.8 for the lower altitude and 0.75 for the higher altitude. Each map has 30 regions, which translates to 60 nodes across two fixed altitudes.

We implemented our approach using Python 3.8. We tested our approach and the baselines on an Ubuntu 20.04 desktop with an AMD Ryzen 9 5950x 3.4 GHz CPU with 16 cores (32 threads) and 126 GB of RAM. We used Gurobi [10] as the mixed-integer linear programming solver to produce our results. We did not modify any of the default parameters set by Gurobi for the solving procedure.

Chapter 5

Results and Discussion

5.1 Information Gain Results

The results from the 10 randomized experiments across each budget for the city environment and the wildlife environment using a team of two agents are shown in Fig. 5.1 and Fig. 5.2. The separation between the green line and the blue line in the plots represents the difference between the total reward in the search space and the maximum amount of reward that agents can obtain given their budget limitations. So, as the budget allocated to the agents increases, the potential reward the agents can receive gets closer to the total reward in the map. We observe that across all budgets, our approach after the early stop and the optimal solution from our approach outperforms all baselines. The greedy baseline performed similar to our approach while the naive greedy and random baselines performed the worst.

Across all tests, the random planner performs the worst because it's unable to leverage the reward structure of balancing high-altitude and low-altitude views to maximize reward. The naive greedy planner performs second worst because it expends a large portion of its budget traveling between the highest information regions within the map. The greedy planner is able to perform similar to our approach as it considers the cost of search and transition actions as part of its reward formulation. This causes the greedy planner to obtain immediate gains in its reward as it takes locally optimal actions. Because our method can optimize over the entire budget of an agent, it produces plans that get the closest to the globally optimal solution.

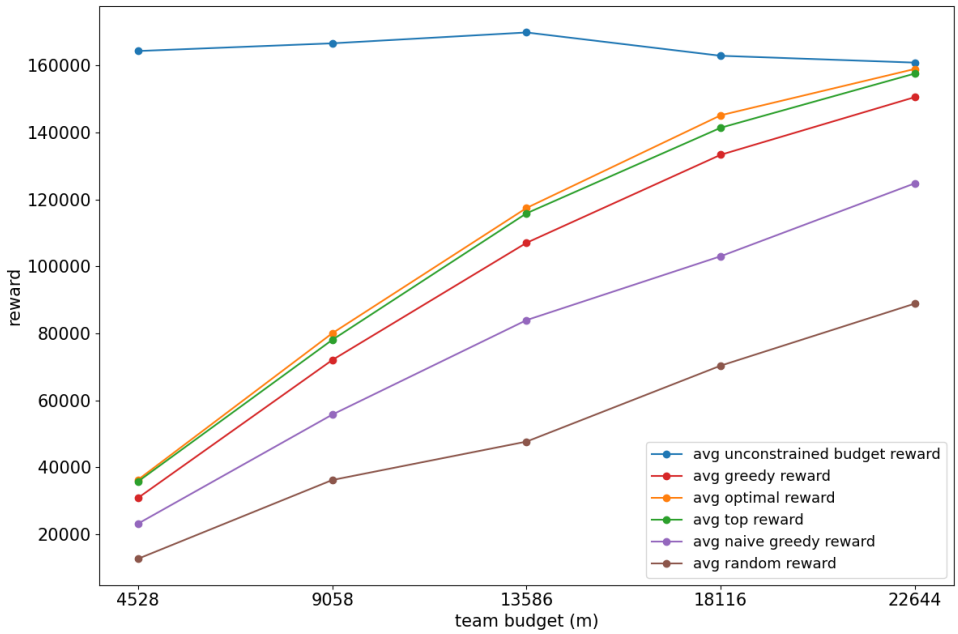

Figure 5.1: Average reward across 10 randomized runs per budget across all baselines for the city environment with a team of two agents

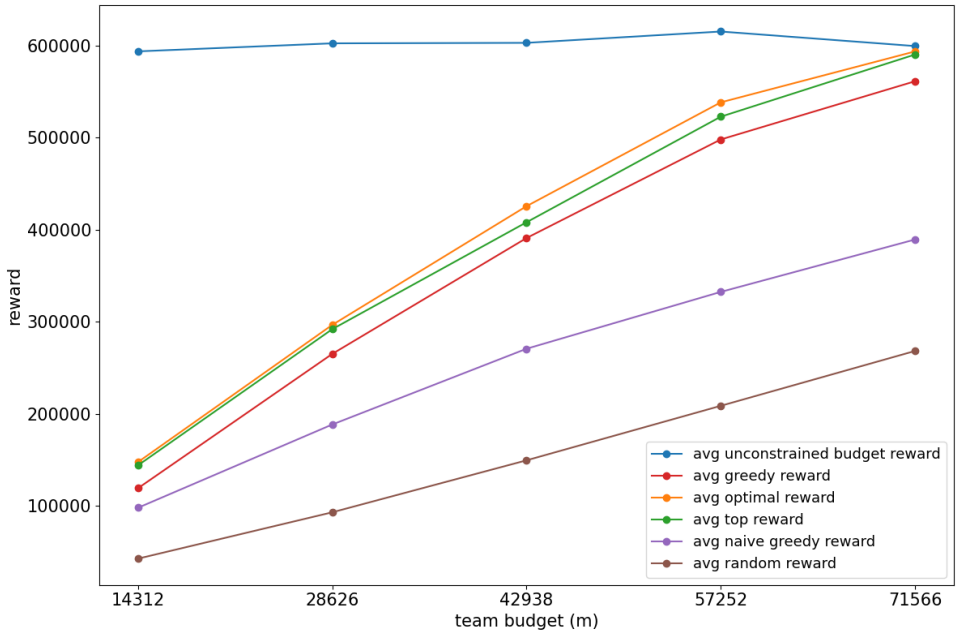

Figure 5.2: Average reward across 10 randomized runs per budget across all baselines for the city environment with a team of two agents

5.2 Suboptimality Results

The percent suboptimality of solutions across each budget for the 10 randomized experiments for each map can be seen in Table 5.1 and Table 5.2. Across all budgets, our approach was able to generate solutions of greater optimality when compared to the baselines. In line with the information gain rewards, the greedy planner performed similar to our approach while the naive greedy, and random planner performed the worst. The overall trend we observe is that as the budget allocated to the agents increases, the gap between our approach after an early stop and the greedy approach gets smaller.

	4528 m	9058 m	13586 m	18116 m	22644 m
TOP Early Stop	1.45	2.47	1.33	2.57	0.83
Greedy	14.95	9.95	8.85	8.09	5.25
Naive Greedy	36.00	30.30	28.55	28.98	21.46
Random	64.93	54.75	59.38	51.48	44.15

Table 5.1: Percent optimality gaps given different budgets for a team of two agents in the city environment

	14312 m	28626 m	42938 m	57252 m	71566 m
TOP Early Stop	2.36	1.61	4.06	2.90	0.57
Greedy	19.15	10.58	8.10	7.52	5.43
Naive Greedy	33.47	36.43	36.34	38.24	34.31
Random	70.87	68.48	64.79	61.20	54.71

Table 5.2: Percent optimality gaps given different budgets for a team of two agents in the wildlife environment

The percent suboptimality of solutions across the various budgets for the city environment is seen in Table 5.1. At the most limited budget, our approach generates plans that are 13.5% more optimal than the greedy planner. At the largest budget, this difference in optimality reduces to 4.42%. For the wildlife environment, the

gap in optimality between our approach and the greedy approach for the lowest budget is 16.79%. The difference in optimality at the largest budget is similar to the city environment at 4.86%. The greedy planner performed worse in the wildlife environment at lower budgets and slightly better than the city environment at larger budgets. While there was a general downward trend in the performance of the naive greedy planner in the city environment, this trend was less consistent for the wildlife environment. Finally, the random planner performed worse in the wildlife environment when compared to the city environment.

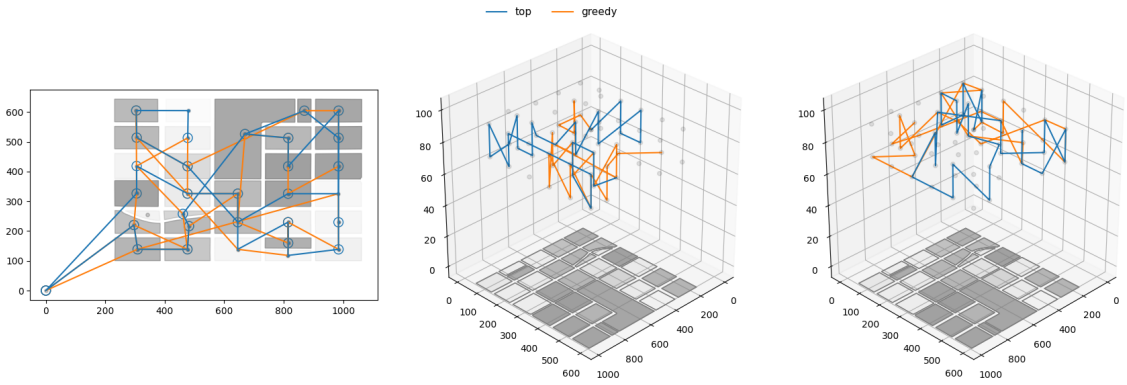

Figure 5.3: Visualization of the paths generated by our approach and the greedy planner for a city environment test case. The blue line is our approach and the orange line is the greedy approach. The alpha values for the regions correspond to the likelihood of objects of interest lying within the region. The plot on the left shows a birds-eye view of the plans generated for the two agents. Open circles are higher altitude measurements and solid circles are lower altitude measurements. The middle and right plots show the paths for the first and second agents respectively. The gap in optimality between our approach and the greedy planner was 2.82% for these plans

The difference in performance across both maps lies in the relationship between cost and reward from the sensor model as well as the spatial distribution of information in each map. For reward, differences in sensing quality dictate the spread of total reward across the higher and lower altitudes. Because the lower altitude has a higher sensing resolution, the majority of reward in the search space will be held by the set of lower altitude nodes. Thus, the improvement in reward diminishes as the budget increases especially when it surpasses the budget needed to complete a coverage over all lower altitude nodes. The second component is cost. For our approach and the

greedy planner, the cost for an action consists of two components: a transition cost and a task completion cost. The transition cost is the budget needed to travel to a node and the task completion cost is the budget needed to perform the search at a given node. The largest proponent of the cost comes from the task completion cost and this is dictated by the fixed sensing altitude the agents fly at.

Given that the sensor model dictates the reward and task completion cost, our experiments observed how task distances and the distribution of information across tasks affect overall system performance. In the city environment, the majority of regions are similar in size and densely grouped. This structure is apt for the greedy planner to succeed as it isn't penalized as much for taking locally optimal actions. We can observe this in Fig 5.3. In the example, we equipped the agents with the estimated coverage budget. The gap in reward across both maps was 2.887%. Although this difference is relatively small, we can qualitatively observe that the paths generated by our approach are smoother than the greedy planner. Because of the diminished reward from a larger budget and the lack of penalty from greedy transitions, the final reward for our approach is close to that of the greedy planner.

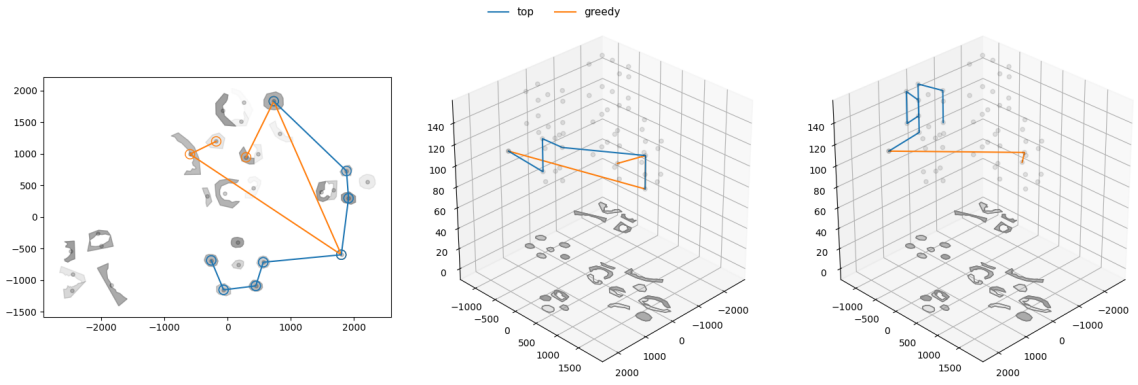

Figure 5.4: Visualization of the paths generated by our approach and the greedy planner for a wildlife environment test case. The blue line is our approach and the orange line is the greedy approach. The alpha values for the regions correspond to the likelihood of objects of interest lying within the region. The plot on the left shows a birds-eye view of the plans generated for the two agents. Open circles are higher altitude measurements and solid circles are lower altitude measurements. The middle and right plots show the paths for the first and second agents respectively. The gap in optimality between our approach and the greedy planner was 28.87% for these plans.

In contrast, the wildlife environment consists of regions that are all different shapes and sizes, and the regions are grouped spatially into clusters. The greedy planner in these environments is penalized more for its locally optimal actions. This can be seen in Fig. 5.4. In this scenario, the agents were given 20% of the estimated budget needed to perform a coverage over the search space. The gap in optimality between our approach at the early stop and the greedy planner was 28.87%. The greedy planner jumps to the regions with the highest reward which causes it to expend a large portion of its budget on transition costs. Our approach on the other hand visits one of the higher reward nodes that greedy visits. But, through performing a tour through a larger set of lower reward nodes, it is able to maximize overall reward.

Thus, although our approach was able to beat the baselines across all budgets in these two maps, it is best suited for scenarios where agents are equipped with a limited budget and the regions are sparsely distributed in the map with varying importance.

5.3 Optimizer Runtime Analysis

Optimizers provide the benefit of being able to produce provably optimal solutions. But, this comes at the trade-off of a long runtime to generate and prove the optimality of a given solution. Greedy planners, on the other hand, provide approximations or even sometimes the optimal solution to a problem in a much quicker time frame. For our tests, the greedy planner returned plans close to instantaneously, so we focused a portion of our analysis on the time-based efficiency of our solver. For the city and wildlife maps, across our 10 experiments per budget, we plotted how fast it took for the solver to receive an equivalent reward to the greedy planner and we plotted the estimated suboptimality gap of the optimal solution we grabbed at the end of our maximum planning time of 1800 seconds.

These plots for the city and wildlife environment can be seen in Fig 5.5 and Fig. 5.6. The left plot for each map plots the time that it took to compute a solution with an equivalent reward to the greedy planner across all budgets and the right plot shows the estimated suboptimality of our optimal solution that we grab at the end of our maximum planning time and the true optimal solution for the problem. Both maps share a similar trend where the amount of time needed to reach the greedy solution

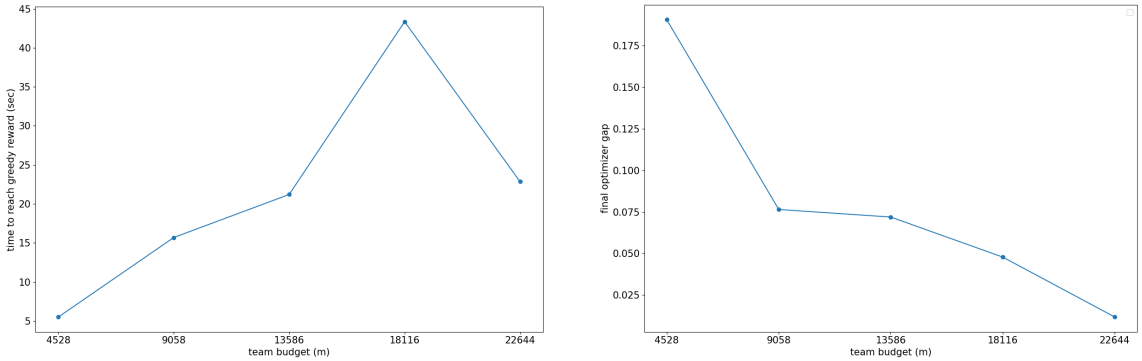

Figure 5.5: Runtime for the early stop spent getting to an equivalent reward as the greedy solution and the final gap at the end of the planning timeout for the city environment

reward is smallest at the lowest budget. It then steadily increases and reaches its peak when agents are equipped with 80% of the estimated coverage budget. Then, this runtime drops when agents are given the full estimated coverage budget. They also share similar trends for the second plot where the estimated gap between the optimal solution we generate and the true optimal solution is highest at the smallest budgets and lowers as the budget allocated to the agents increases.

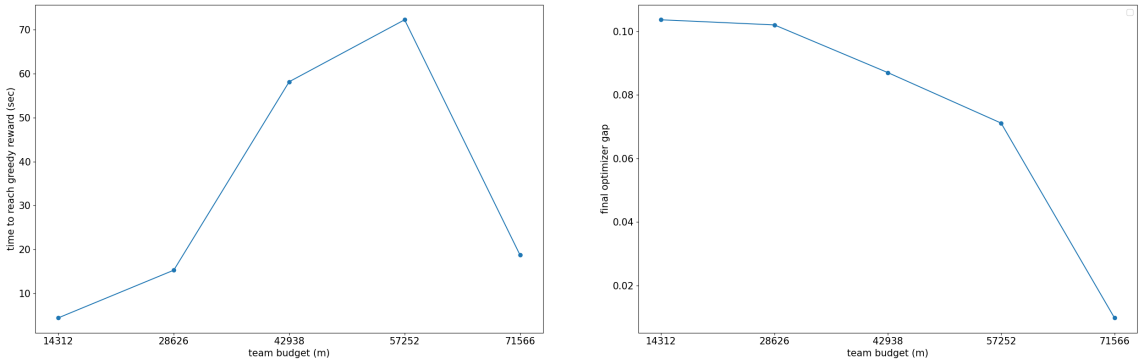

Figure 5.6: Percentage of runtime for the early stop spent getting to an equivalent solution as the greedy solution and the final gap at the end of the planning timeout for the wildlife environment.

At the lowest budget, the search space for the planning problem is smaller. So, the solver can quickly find a solution with an equivalent reward to the greedy planner and consequently spends the majority of the remaining runtime on improving the

reward of its solution beyond that of the greedy reward. As the budget gets larger, the problem gets more complex. So, the solver takes longer to find a solution with an equivalent reward to the greedy solver. Finally, when the agents are given the estimated coverage budget, the problem is easier as the majority of the reward in the space can be collected, so the solver's performance improves.

5.4 Variable Team Size

For our experiments, we also ran various tests to observe how the performance of our system changes as we vary the size of the agent team. We ran 10 randomized experiments per budget for a team size of one and three agents. The resulting optimal solutions generated by our solver after the gap threshold or maximum runtime are shown in Fig. 5.7 for both maps. We observe that for most budgets the one agent team produces plans with a higher reward than the two and three agent team. Although the one agent team is able to generate plans with the highest reward, this comes at the tradeoff of runtime since it takes twice as long as the two agent team and three times as long as the three agent team to generate this reward.

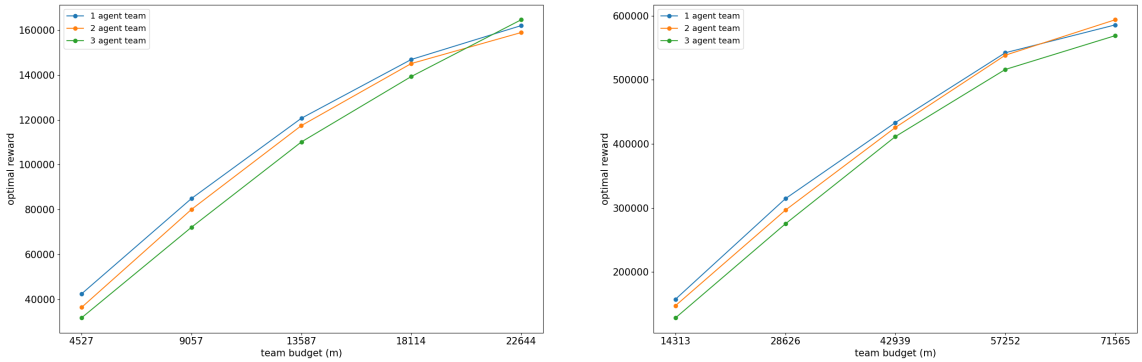

Figure 5.7: Optimal reward obtained by the solver for variable agent team sizes given the gap threshold and the maximum runtime. The left plot shows the results for the city environment and the right plot shows the results for the wildlife environment.

The associated average rewards for our approach and the baselines for the city and wildlife map can be seen in Fig. 5.8 and Fig. 5.9 respectively. At a high-level, we observe similar trends in performance to the two agent tests in Fig. 5.1 and Fig. 5.2 respectively. The suboptimalities for these plots can be seen in Table 5.3 and Table

Team Size	Approach	4529 m	9058 m	13586 m	18116 m	22644 m
1	TOP Early Stop	0.15	0.12	0.25	0.55	0.0
	Greedy	11.62	9.29	10.80	8.75	4.71
2	TOP Early Stop	1.45	2.47	1.33	2.57	0.83
	Greedy	14.95	9.95	8.85	8.09	5.25
1	TOP Early Stop	0.65	2.37	3.07	3.22	1.83
	Greedy	8.35	10.79	9.51	7.39	6.61

Table 5.3: Optimality gaps given different budgets for a variable team of agents in the city environment

5.4. We focus our analysis on our early stop solution and the greedy planner. For the city environment, the optimality of solutions for our approach were better when increasing the team size from one to two agents and decreases with the team size of three agents. For the wildlife environment, the optimality is consistently better with the two and three agent team when compared to the single agent team.

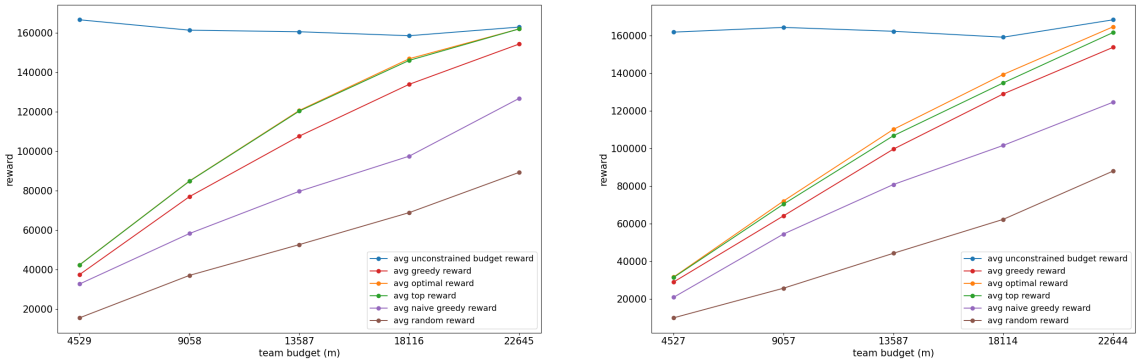

Figure 5.8: Average reward across 10 randomized runs per budget across all baselines for the city environment with a team of one and three agents agents.The left plot shows the results for the one agent team size and the right plot shows the results for the three agent team size.

We measure the efficiency of the solver when scaling to larger team sizes by observing the estimated suboptimality of our optimal solution that we grab at the end of our maximum planning time and the true optimal solution for the problem. The results from these experiments can be seen in Fig. 5.10. Similar to the two agent tests, the estimated gap is largest at the lowest budget and then decreases as the budget allocated to the agents increases. If we compare performance across the various team sizes, the estimated gap increases as we increase the team size.

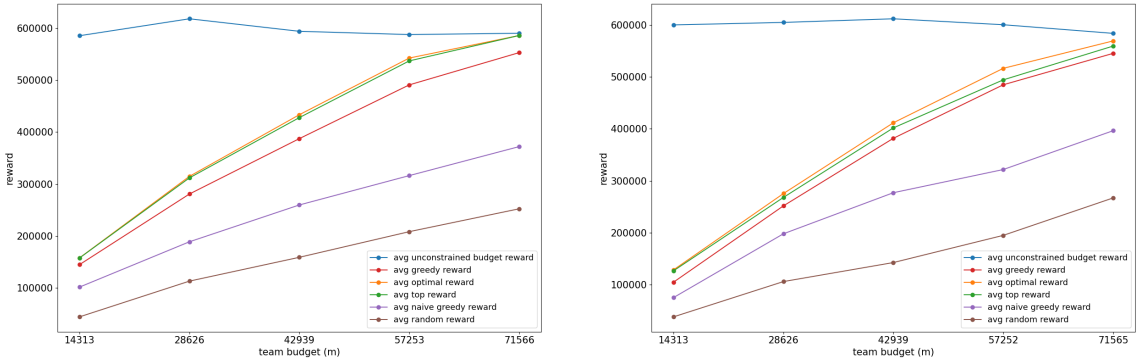

Figure 5.9: Average reward across 10 randomized runs per budget across all baselines for the city environment with a team of one and three agents. The left plot shows the results for the one agent team size and the right plot shows the results for the three agent team size.

Team Size	Approach	14313 m	28626 m	42939 m	57253 m	71566 m
1	TOP Early Stop	0.00.11	0.00.91	0.01.27	0.01.01	0.0
	Greedy	0.08.26	0.10.81	0.10.62	0.09.48	0.05.68
2	TOP Early Stop	0.02.36	0.01.61	0.04.06	0.02.90	0.00.57
	Greedy	0.19.15	0.10.58	0.08.10	0.07.52	0.05.43
1	TOP Early Stop	0.01.46	0.02.67	0.02.37	0.04.26	0.01.72
	Greedy	0.1841	0.0840	0.0723	0.0612	0.0423

Table 5.4: Optimality gaps given different budgets for a variable team of agents in the wildlife environment

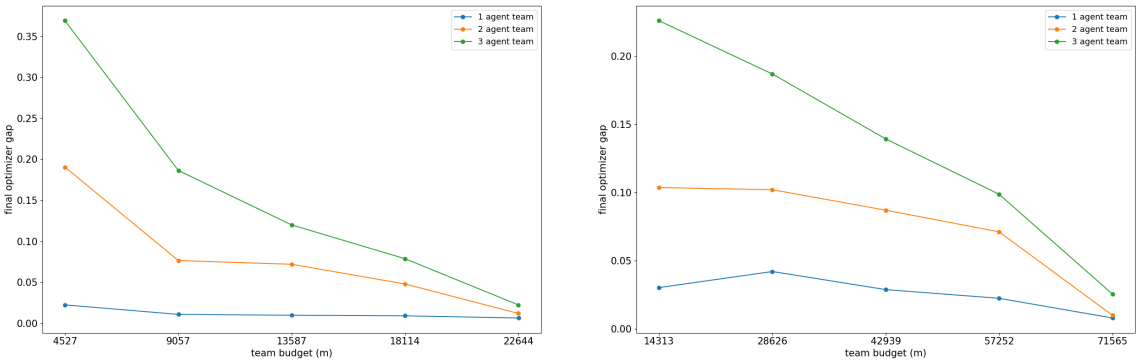

Figure 5.10: Final gap at the end of the planning runtime or after the gap threshold was reached across the city and wildlife environment for variable team sizes. The plot on the left shows the results for the city environment and the plot on the right shows the results for the wildlife environment.

In summary, when increasing the team size, the difference between the optimal reward generated from the maximum planning timeout and the one agent increases. As the number of agents in the team increases, the number of search variables for the problem increases exponentially. Thus, the solver struggles to efficiently search this increasingly larger space as the number of agents in the team increases. Despite these challenges, our approach is still able to outperform the baselines.

Chapter 6

Conclusions and Future Work

In this thesis, we present a multi-agent informative path planning approach that allows small teams of agents to balance the trade off between sensing quality and budget expended when tasked with searching over a set of regions. Through our experiments, we're able to show a consistent improvement over the baselines when agents are allocated a limited sensing budget. Thus, agents are able to balance the tradeoff between revisiting areas of high importance and exploring new regions to generate plans with greater optimality.

Future directions of this work could include speeding up the optimization process. This could be done by feeding a warm start to the planner using the greedy planner or by developing user-defined heuristics to aid the solver. Another avenue for potential future work is picking the right parameters to weight the search across areas and roadways based on the user's desired behavior for the system. Additionally, modfiications to the base formulation could be made to account for uncertainty and to allow for partial costs and rewards for regions. Our method is currently centralized, so additional future work could include extending this approach to a distributed and decentralized case.

Bibliography

[1] Sankalp Arora and Sebastian Scherer. Randomized algorithm for informative path planning with budget constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4997–5004, 2017. doi: 10.1109/ICRA.2017.7989582. 2.1, 2.3

[2] Sankalp Arora, Sanjiban Choudhury, and Sebastian Scherer. Hindsight is only 50/50: Unsuitability of mdp based approximate pomdp solvers for multi-resolution information gathering, 2018. URL https://arxiv.org/abs/1804.02573. 2.3

[3] Rafael Bailon-Ruiz, Arthur Bit-Monnot, and Simon Lacroix. Real-time wildfire monitoring with a fleet of uavs. *Robotics and Autonomous Systems*, 152:104071, 2022. 1.1

[4] Jonathan Binney and Gaurav S Sukhatme. Branch and bound for informative path planning. In *2012 IEEE international conference on robotics and automation*, pages 2147–2154. IEEE, 2012. 2.1

[5] Jonathan Binney, Andreas Krause, and Gaurav S. Sukhatme. Informative path planning for an autonomous underwater vehicle. In *2010 IEEE International Conference on Robotics and Automation*, pages 4791–4796, 2010. doi: 10.1109/ROBOT.2010.5509714. 2.1

[6] Jonathan Binney, Andreas Krause, and Gaurav S Sukhatme. Optimizing waypoints for monitoring spatiotemporal phenomena. *The International Journal of Robotics Research*, 32(8):873–888, 2013. 1.1

[7] Gianni A. Di Caro and Abdul Wahab Ziaullah Yousaf. Multi-robot informative path planning using a leader-follower architecture. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10045–10051, 2021. doi: 10.1109/ICRA48506.2021.9561955. 2.2

[8] Ayan Dutta, Anirban Ghosh, and O. Patrick Kreidl. Multi-robot informative path planning with continuous connectivity constraints. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3245–3251, 2019. doi: 10.1109/ICRA.2019.8794090. 2.2

[9] Ayan Dutta, Amitabh Bhattacharya, O Patrick Kreidl, Anirban Ghosh, and Prithviraj Dasgupta. Multi-robot informative path planning in unknown environments through continuous region partitioning. *International Journal of Advanced Robotic Systems*, 17(6):1729881420970461, 2020. doi: 10.1177/1729881420970461. 2.2

[10] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL https://www.gurobi.com. 4.3

[11] Gregory Hitz, Enric Galceran, Marie-Ève Garneau, François Pomerleau, and Roland Siegwart. Adaptive continuous-space informative path planning for online environmental monitoring. *Journal of Field Robotics*, 34(8):1427–1449, 2017. 1.1

[12] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based motion planning for robotic information gathering. In *Robotics: Science and Systems*, volume 3, pages 1–8, 2013. 2.1

[13] Stephanie Kemna, John G. Rogers, Carlos Nieto-Granda, Stuart Young, and Gaurav S. Sukhatme. Multi-robot coordination through dynamic voronoi partitioning for informative adaptive sampling in communication-constrained environments. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2124–2130, 2017. doi: 10.1109/ICRA.2017.7989245. 2.2

[14] Ariella Mansfield, Sandeep Manjanna, Douglas G. Macharet, and M. Ani Hsieh. Multi-robot scheduling for environmental monitoring as a team orienteering problem. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6398–6404, 2021. doi: 10.1109/IROS51168.2021.9636854. 2.2

[15] Clair E Miller, Albert W Tucker, and Richard A Zemlin. Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4): 326–329, 1960. 3.6

[16] Brady Moon, Satrajit Chatterjee, and Sebastian Scherer. Tigris: An informed sampling-based algorithm for informative path planning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 19, page 5760–5766. IEEE, October 2022. doi: 10.1109/iros47612.2022.9981992. URL http://dx.doi.org/10.1109/IROS47612.2022.9981992. (document), 2.1, 3.4

[17] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org, 2017. 4.1

[18] Liam Paull, Carl Thibault, Amr Nagaty, Mae Seto, and Howard Li. Sensor-driven area coverage for an autonomous fixed-wing unmanned aerial vehicle. *IEEE transactions on cybernetics*, 44(9):1605–1618, 2013. 3.4

[19] Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Jen Jen Chung, Inkyu Sa, Roland Siegwart, and Juan Nieto. An informative path planning framework for

uav-based terrain monitoring. *Autonomous Robots*, 44(6):889–911, 2020. 2.3

[20] Seyed Abbas Sadat, Jens Wawerla, and Richard Vaughan. Fractal trajectories for online non-uniform aerial coverage. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2971–2976, 2015. doi: 10.1109/ICRA.2015.7139606. 2.3

[21] Lukas Schmid, Michael Pantic, Raghav Khanna, Lionel Ott, Roland Siegwart, and Juan Nieto. An efficient sampling-based method for online informative path planning in unknown environments. *IEEE Robotics and Automation Letters*, 5 (2):1500–1507, 2020. doi: 10.1109/LRA.2020.2969191. 2.1

[22] Yunfei Shi, Ning Wang, Jianmin Zheng, Yang Zhang, Sha Yi, Wenhao Luo, and Katia Sycara. Adaptive informative sampling with environment partitioning for heterogeneous multi-robot systems. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11718–11723, 2020. doi: 10.1109/IROS45743.2020.9341711. 2.2

[23] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34: 707–755, April 2009. ISSN 1076-9757. doi: 10.1613/jair.2674. URL http://dx.doi.org/10.1613/jair.2674. 2.1

[24] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. *European Journal of Operational Research*, 209 (1):1–10, 2011. 3.6

[25] Jingjin Yu and Steven LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 27, pages 1443–1449, 2013. 1.2

[26] Jingjin Yu, Mac Schwager, and Daniela Rus. Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 342–349, 2014. doi: 10.1109/IROS.2014.6942582. 2.1

[27] Alexander Zelinsky, Ray A Jarvis, JC Byrne, Shinichi Yuta, et al. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *Proceedings of international conference on advanced robotics*, volume 13, pages 533–538. Citeseer, 1993. 3.3

[28] Hai Zhu, Jen Jen Chung, Nicholas R.J. Lawrance, Roland Siegwart, and Javier Alonso-Mora. Online informative path planning for active information gathering of a 3d surface. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1488–1494, 2021. doi: 10.1109/ICRA48506.2021.9561963. 1.1