PRIORITIZED ROBOTIC EXPLORATION WITH DYNAMIC DEADLINES

by

Sayantan Datta

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2024

Approved by:

Dr. Srinivas Akella

Dr. Min Shin

Dr. Erik Saule

Dr. James M. Conrad

ABSTRACT

SAYANTAN DATTA. Prioritized Robotic Exploration with Dynamic Deadlines.
(Under the direction of DR. SRINIVAS AKELLA)

Autonomous exploration using mobile robots, commonly referred to as robotic exploration, entails simultaneously performing robot perception, localization, and motion planning to explore an unknown environment. Most prior indoor robotic exploration algorithms focus on exploring the entire environment. We consider exploration under deadlines dynamically imposed either by the robot's battery or by the environment. Such time-sensitive robotic exploration is critical in dangerous environments as it provides vital initial information about the geometric structure and layout of the environment for subsequent operations. For instance, firefighters can utilize an initial map generated by this deadline constrained robotic exploration to rapidly navigate a building on fire. In the presence of deadlines, the robots should identify the semantically significant regions of the environment (e.g., corridors) and prioritize those that enable them to determine the environment's geometric structure and return to the starting position before the deadline.

This dissertation addresses the problem of autonomous exploration in indoor environments with dynamic deadlines. The problem is NP-hard and requires exponential time to solve optimally. Therefore, we present a short-horizon exploration algorithm, the priority-based greedy exploration algorithm, and several long-horizon exploration algorithms; these include adaptations of the orienteering problem and the profitable tour problem for single-robot and multi-robot exploration of unknown environments with dynamic deadlines. Furthermore, we present a test suite of environments and exploration metrics to benchmark the real-world efficiency of exploration algorithms in office-like environments. Our single-robot experiments reveal that the priority-based greedy exploration algorithm, which focuses on exploring semantic regions with higher connectivity, consistently outperforms the baseline cost-based greedy exploration algorithm in terms of environment layout identification and exploration efficiency. The priority-based greedy algorithm was found to be on par with the computationally expensive long-horizon exploration algorithms in terms of percent of the area explored within the deadline. Long-horizon exploration algorithms on the other hand exhibit consistent performance with low variance over repeated experiments. Moreover, the multi-robot priority-based greedy exploration algorithm demonstrated better performance compared to the multi-robot baseline exploration algorithm and performed on par with the multi-robot long-horizon based exploration algorithm while being computationally faster.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor, Dr. Srinivas Akella, for his guidance, support, and encouragement throughout my doctoral studies. He has been a great mentor and a source of inspiration for me. I have learned a lot from his expertise, insights, and vision. I am also grateful to him for providing me with excellent research opportunities and facilities.

I would also like to thank my colleagues, Saurav Agarwal and Kalvik Jakkala, for their friendship, collaboration, and assistance. They have been very helpful and supportive in various aspects of my research. I have enjoyed working with them and sharing many memorable moments. I am thankful to my lab peers, David Vutetakis, Yupeng Yang, Yanze Zhang, Ian Gao, and Siwon Jo for their valuable feedback, suggestions, and discussions. Thank you, everyone, for maintaining positivity even after successive failed experiments. I extend sincere thanks to the undergraduate students I mentored and worked with: Jacob Dent, Philip Smith, Adam Hudson, Sam Crane, Lauren Wylie, and Erich Choudhury, for their valuable feedback, suggestions, and discussions.

I appreciate the help and advice of my mentors, who have guided me throughout my academic journey and career development. I would like to acknowledge Dr. Sterling McLeod, and Dr. Wenhao Luo, who have been very generous with their time, equipment, and knowledge. Sincere thanks also to Dr. Dipankar Maity for graciously loaning me his robots at a moment's notice, enabling me to test my algorithms. Additionally, I would like to offer my sincere thanks to Jan Mikula from the Czech Technical University in Prague for his insightful comments and assistance in formulating my research.

I am indebted to my committee members, Dr. Erik Saule, Dr. Min Shin, and Dr. James Conrad, for their constructive comments and suggestions on my dissertation. They have not only provided me with useful insights and perspectives on my research topic but have also mentored me throughout this process. They have been readily available for discussions without the need for formal appointments and have actively participated in brainstorming sessions to refine my ideas. I am honored to have them as my committee members.

On a personal note, I would like to express my deep gratitude to my family for their financial support throughout my academic career. I would like to express my heartfelt thanks to my friends, Antardipan Pal, and Madhumita Paul, for making life outside the lab something to look forward to. They have been my pillars of strength and support throughout this journey. They have always been there for me, cheering me up and motivating me. I would also thank Pushpita Ghosh for proofreading my thesis. Last but not the least, I would like to immensely thank my girlfriend, Rittika Mallik, who has been my constant companion. She has been my biggest fan and my greatest critic. She has always encouraged me to pursue my dreams and goals. She has made me a better person and a happier one.

DEDICATION

Dedicated to Rittika, who taught me never to give up.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

Robotic exploration enables autonomous navigation in environments that are either unsuitable or hazardous for human presence. This technology facilitates collaborative exploration with human teams, augmenting their capabilities in various applications, such as mapping perilous indoor settings and conducting search and rescue operations. Currently, most robotic exploration techniques [1, 2, 3, 4] predominantly aim at complete exploration of the initially unknown environment as quickly as possible, often neglecting the constraints imposed by the robot's operational time limits or environmental factors.

As mobile robots run on batteries, the battery life of the robots is limited. This limitation specifically applies to unmanned aerial vehicles (UAVs), especially quadcopters. For example, a DJI Phantom 4 has a battery lifespan of 25 minutes, while an Unmanned Ground Vehicle (UGV) such as Turtlebot3 has a battery lifespan of about 2 hours. In practice, the operational time, the time during which the robot doesn't start flashing low battery sign, of the robots are much lower than the battery lifespan. Environmental conditions can further reduce this operational time. Two illustrative scenarios where environmental factors significantly influence the operational time are presented:

- **Building on fire:** In this scenario, a robot equipped with LiDAR and sonar sensors is deployed to navigate a burning building. The objective is to map the building's layout and provide it to firefighters before their entry. The robot is assigned a deadline that considers both the urgency of the firefighters' need for the layout and the extent of damage the fire could inflict on the robot. The robot must complete its exploration and return to the operator within this stipulated deadline.

- **Radioactive environment:** In the event of a radiation leak at a nuclear power plant, human entry even with safety equipment becomes prohibitive. A robot is then tasked with exploring the facility. Due to the thick walls impeding wireless communication, the robot must return safely to the exit to relay its collected data. As the robot navigates the nuclear plant, depending on the radiation levels, the robot gets a limited time to explore the environment before the radiation starts affecting the robots sensors and controls. The robot is required to complete its exploration within this constrained deadline, adapting to any unexpected changes of its deadline.

This dissertation introduces and discusses several methods by which a robotic system can explore an environment within a dynamic deadline. The robotic system is constrained by a limited time to explore the environment, necessitating a prioritized exploration of the environment to ensure that the most critical segments of the environment's layout are mapped before the deadline. The robotic system may receive multiple updated deadlines, reflecting changes in environmental conditions or the non-linear depletion of the robot's battery.

We introduce several prioritized exploration algorithms for a single robot exploring an indoor environment with a dynamic deadline. The first prioritized exploration algorithm is a priority-based greedy algorithm that works in an iterative greedy fashion. sdThe priority-based greedy algorithm selects the visiting the locally optimal exploration target with a one step lookahead. An exploration target is a location in the explored environment that the robot visits expecting to view an unexplored part of the environment. We investigate alternative prioritized exploration algorithms in an attempt to improve the priority-based greedy solution by modeling the problem as an Orienteering problem, a Profitable Tour problem, a Minimum Latency Paths problem, and a Profitable Tour problem with Minimum Latency paths. We find that the priority-based greedy solution provides us with competitive results with short deadlines while being outperformed by other formulations for longer deadlines. We compare the algorithms in simulation and real world experiments. We provide a detailed discussion analyzing these results.

Furthermore, we extend our investigation to a multi-robot variant of the prioritized exploration problem. We introduce two exploration algorithms: a multi-robot priority-based greedy algorithm and a multi-robot profitable tour problem based exploration algorithm. We develop a heuristic based on the team orienteering problem to solve the multi-robot profitable tour problem based exploration algorithm. The performance of these algorithms is compared through simulations, accompanied by a discussion of the findings.

This dissertation presents the theory, algorithms, simulation and real-world experiments of the prioritized exploration algorithms with dynamic deadlines. We develop exploration metrics to assess the quality of our algorithms and statistical tests to ascertain their significance. Chapter 2 introduces the single-robot prioritized exploration problem with dynamic deadlines and presents greedy solutions to the problem. Chapter 3 introduces optimization based algorithms for the single-robot prioritized exploration problem. Finally, Chapter 4 introduces the multi-robot variant of the prioritized exploration problem with dynamic deadlines and presents our approach and solutions. The dissertation concludes in Chapter 5, where we summarize our research contributions and discuss several directions for future work.

CHAPTER 2: PRIORITIZED INDOOR ROBOT EXPLORATION WITH DYNAMIC DEADLINES

In robotic exploration, the robot starts exploring a initially unknown environment to explore the environment till the exploration is complete or till other factors such as robot battery and environmental constraints inhibit the robot from exploring any further. The robot senses the environment with its sensors and represents the obstacles and free regions as a map. It takes a decision to visit the next position at the periphery of the obstacle free region and the unmapped region on the map. As the robot is unaware of the remaining unexplored environment, it lacks foresight into what structure and connectivity of environment it can expect while exploring. Therefore, it takes a decision based on the current known map of the environment.

As the robot explores the environment, the environment or the robot's battery imposes a deadline on the exploration time. The robot should come back to the starting location of the exploration before the imposed deadline. If multiple deadlines are sequentially imposed on the robot, the robot must explore according to the updated deadline. The goal of the exploration is to determine the geometric structure of the unknown environment as rapidly as possible and return to the home location. With the imposed constraints, the robot should prioritize certain parts of the environment to ensure it is able to determine the geometric structure before the deadline.

An iteration of the robotic exploration problem starts with the robot equipped with sensors to observe the environment around it and simultaneous localization and mapping algorithms to localize itself to the partially explored environment it is in. The robot determines a set of frontier points. Each frontier point is a location to travel to on the explored map from which the robot expects to view a portion of the unexplored region. The robot fixes on one particular frontier point to travel to based the exploration algorithm. This specific frontier point is referred to as the target point. This iteration of the robotic exploration ends when the robot navigates to the chosen frontier point.

2.1 Problem Statement

A single robot explores an unknown environment and models the environment as an exploration graph, $G = (V, E)$. V is the set of explored vertices, where each vertex $v \in V$ represents a point in the exploration environment. E represents the set of edges, where $e_{ij} \in E$ is an edge that connects two vertices v_i and v_j. An edge represents a collision free path between the two vertices. Each edge has a weight w_{ij} which is the estimated time to traverse the path represented by the edge. The set of vertices, V can be divided into two subsets, discovered vertices V_d and visited vertices V_{vis}.

The robot selects a target vertex $v_t \in V_d$ to visit. On visiting the vertex, the robot explores a part of the unknown environment. Once the map is updated, a new exploration graph G is formulated based on the new explored environment.The objective of the robot is to explore the graph G efficiently by identifying the layout of the environment within a deadline t_{r_0} imposed by the environment or the robot's battery. The deadline t_{r_0} is imposed after a time period t_a, when the robot explores without a deadline. The value of t_a is unknown to the robot. The robot is unaware of the values of t_a and t_{r_0} at the start of the exploration. Neither does it know when the deadline will be imposed nor what exploration time it would have when the deadline is imposed. The timeline of robot exploration is illustrated in Figure 2.1.

2.2 Related work

Autonomous robotic exploration is an important and well-studied problem. A widely used approach for robotic exploration is frontier-based exploration, which

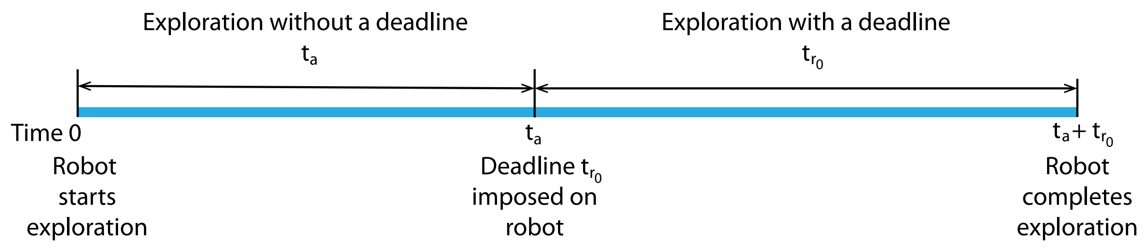

Exploration without a deadline
t_a
Exploration with a deadline
t_{r_0}
Time 0
Robot starts exploration
t_a
Deadline t_{r_0} imposed on robot
$t_a + t_{r_0}$
Robot completes exploration

Figure 2.1: Timeline of robot exploration. The exploration starts at time 0, when it explores unaware of a deadline. After t_a timesteps, a deadline of t_{r_0} is imposed on the robot. The robot is aware of the current deadline t_{r_0} . The total exploration time of the robot is $t_a + t_{r_0}$

greedily directs robots to regions likely to provide new information about the environment. Such approaches typically develop occupancy grid representations of environments using either a single robot [1] or multiple robots [5]. Decision theoretic approaches for exploration consider the information gain of exploration actions and robot pose uncertainty reduction of loop-closing actions [6]. See Thrun et al. [7] for an overview of probabilistic mapping techniques that compute occupancy grids using Simultaneous Localization and Mapping (SLAM) approaches.

Robotic exploration was modeled as graph construction by Dudek et al. [8] for a robot that uses portable markers instead of distance and orientation sensors. An exploration and mapping strategy using a topological model based on a spatial semantic hierarchy was developed by Kuipers and Byun [9]. Cowley et al. [10] advocate an efficient exploration approach that uses both topological and metric data to identify places with semantic significance for exploration. The idea was improved by Wang et. al. [3] by using a topological map and semantic information from RGBD sensors to improve indoor exploration. The technique uses semantic information from the environment to distinguish between rooms and corridors. Since their exploration algorithm is directed towards rooms, the robot revisits corridors several times. This makes it less suitable for exploration in environments with dynamic deadlines. Graph-based exploration for subterranean environments was addressed in [11], where the authors implement a two-stage local and global planner that enables the UAV to return to its starting location based on the current remaining flight time and information that is available from the start of the exploration. Similar graph-based exploration in an subterranean environment was also addressed by Bayer et. al. [12, 13]. However, these approach does not consider dynamic deadlines.

Time-limited exploration has been investigated recently [14] using a reinforcement learning based exploration method, and uses a fixed deadline at the start of exploration. Communication and energy constraints during exploration have been addressed by [15] for a multi-robot setup. Energy constraints serve as deadlines; however the focus is on multi-robot coordination, and the robots do not return to their home locations. Semantic information-based coordinated multi-robot exploration has been introduced in [2]. Prior information, in the form of rough topometric graphs provided by a user [16], or predictions of unexplored regions of the environment based on environment structure and a library of previously seen maps [17], has recently been leveraged for more efficient exploration. Semantic information has also been used for task adaptation [18]. Recent work uses deep reinforcement learning (DRL) to learn exploration information over office blueprints [19]. DRL-based exploration [20, 19] has not considered deadlines.

2.3 Greedy Algorithm for Prioritized Robotic Exploration

The greedy algorithm focuses on selecting a specific target vertex v_t depending on whichever vertex is the most rewarding at that time instant. It does so by following a three stage exploration strategy which starts when the robot has a partially explored map and needs to decide a particular location to visit, and ends when the robot has visited the specific location. Figure 2.2 illustrates the three stage solution of the exploration strategy. The first stage maps (Section 2.3.1) the environment using the on-board sensors to create a occupancy map. This occupancy map is converted to a skeleton graph. This skeleton graph is used as the exploration graph, where vertices refer to a local region within the explored area. The second stage of the algorithm

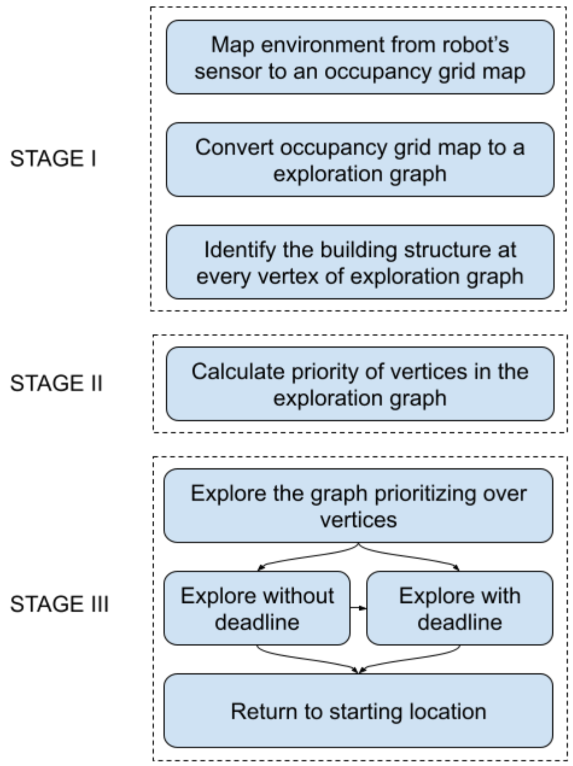

STAGE I
Map environment from robot's sensor to an occupancy grid map
Convert occupancy grid map to a exploration graph
Identify the building structure at every vertex of exploration graph
STAGE II
Calculate priority of vertices in the exploration graph
STAGE III
Explore the graph prioritizing over vertices
Explore without deadline
Explore with deadline
Return to starting location

Figure 2.2: The three-stage exploration strategy developed for single agent prioritized exploration with a dynamic deadline. The first stage converts the explored environment into a graph representation. The second stage calculates priority of candidate locations for the robot to visit. These potential locations are mapped as vertices in the exploration graph. The third stage involves exploring the environment while prioritizing over vertices. It uses two exploration algorithms to explore the environment before returning to the starting location. The first exploration algorithm is used when the deadline is not imposed, while the second is used when the deadline is imposed.

(Section 2.3.2) assigns priorities depending on the structure of the local region. For example, corridors have higher priority than small rooms, as exploring corridors may lead to discovering more rooms. Finally, in the third stage (Section 2.3.3), we have two greedy algorithms, one that is used to explore when the robot does not have a deadline, and second is used when the robot has a deadline imposed on it.

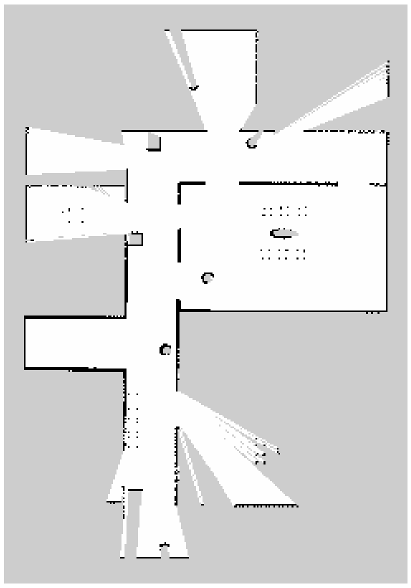
2.3.1 Stage I: Creating the Exploration Graph

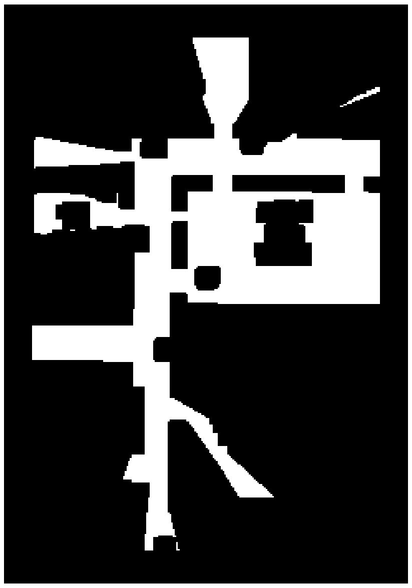
For the graph environments, we assume that the robot discovers the vertices along with their semantic information. The semantic information is the type of region the vertex is in (e.g., corridor, room). We also assume that two vertices are connected by an edge if they have mutual line-of-sight visibility. These simplifying assumptions allow greater focus on the exploration algorithms.

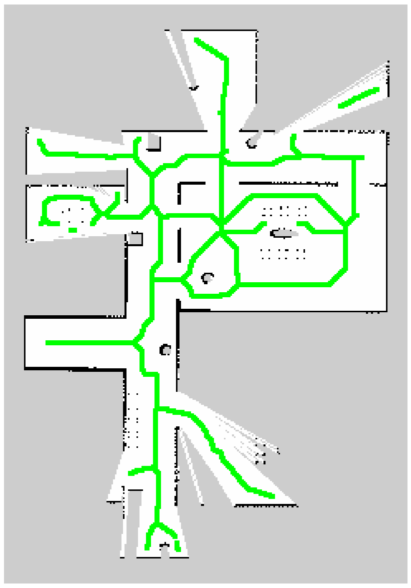
We next discuss creation of the exploration graphs for Gazebo environments. The agent senses an initially unknown environment using its LiDAR sensor. By running a SLAM algorithm on the robot odometry and LiDAR data, we create an occupancy grid map [21] of the environment. Each cell of this occupancy grid shows the probability of it being occupied. This occupancy grid map is morphed into a skeleton image [22]. This image is further transformed into a skeleton graph, which serves as the exploration graph G. The vertices V in this skeleton graph indicate two types of locations of interest in the environment. The first is a frontier region between the explored and unexplored regions. The second is an intersection or branching of possible paths that the robot can take, such as an intersection of two corridors or a corridor and a room. An edge connects a pair of vertices if there is a collision-free path between them. As the robot traverses the environment, the exploration map is updated. A new skeleton graph is generated after the robot reaches the next vertex during exploration. The steps of creating the skeleton graph is illustrated in Figure 2.3.

Our exploration algorithm can also be used, with minimal modifications, with other 2D and 3D environment representations such as topometric graphs [10] and OctoMap [23].

Each vertex can represent the local environment and its associated semantic information observed by the robot's sensor at that position. The shape and size of the available floor region are used to assign a label to a given vertex. For example, if a vertex is in a region that is a narrow passageway or a passageway that leads to different doors, the vertex is labeled as a "corridor". Similarly, if a vertex is in a room with small dimensions, such as a personal office or a closet, it is labeled as a "small room". On the other hand, if a vertex is in an area with large open spaces, it is labeled


(a) Occupancy Map


(b) Map with dilated obstacles


(c) Skeleton image

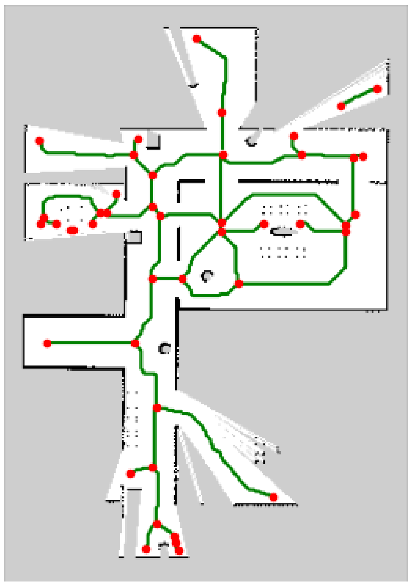

(d) Exploration Graph

Figure 2.3: Stages of converting occupancy grid map to an exploration graph using skeletonization. (a) Shows the initial occupancy map generated during exploration. (b) shows the map with dilated obstacles to ensure the robot doesn't collide with obstacles. (c) shows the skeletonized image of the dilated map. This ensures robot takes safe paths when following the skeleton image. (d) shows the skeleton graph which is used as a the exploration graph G. Here each vertex is shown as a red circle and the edges are shown as green lines.

as a “large room”.

We have implemented a geometry-based classification of corridors, small rooms, and large rooms. For every vertex, we extract a local map, a 3 m $\times$ 3 m window of the occupancy grid map centered on the vertex. This local map provides geometrical information on the obstacles and free space of the region around the vertex. We employ a geometry-based approach to identify the location and orientation of the obstacles in the local map. The vertices are classified based on the number of obstacles, and the distance and orientation of the obstacles relative to one another.

2.3.2 Stage II: Vertex Prioritization

We introduce two exploration algorithms, Priority-based exploration algorithm, abbreviated as P-Greedy and Cost-based exploration algorithm, abbreviated as C-Greedy. Both agents explore an unknown indoor environment, and if time permits, return to the home location. However, they order their visits to discovered vertices differently.

The Priority-based greedy exploration algorithm assigns a level of priority for each type of discovered vertex region. The priority depends on the connectivity of the type of environment the vertex is physically present in. Assuming that corridors connect different parts of a building, they are assigned a higher priority over rooms. Similarly as a large room can connect to other rooms, large rooms are assigned a higher priority over small rooms. The exploration algorithm greedily chooses to visit the vertex v_i with the highest priority, breaking ties by selecting the lower cost vertex. The goal of this algorithm is to rapidly compute the building layout by prioritizing the exploration of corridors over other regions.

Alternatively, the Cost-based greedy exploration strategy explored the nearest vertex first. The algorithm greedily chooses the vertex with the minimum path cost. This algorithm is modeled on Yamauchi’s exploration algorithm [1]. The algorithm considers that all vertices have the same priority.

Algorithm 1: Prioritized Exploration with a Dynamic Deadline

```
Input: starting vertex: $v_0$; deadline: $t_r \leftarrow$ unknown
Result: Explored graph $G$
1  $G \leftarrow \emptyset$ ;                                  // initialize exploration graph
2  $V_d \leftarrow \emptyset$ ;                                 // discovered and unvisited vertices
3  $V_{vis} \leftarrow \{v_0\}$ ;                               // visited vertices
4  Add ($v_0$, E($v_0$)) to $G$ ;   // E($v_i$) is set of edges incident on vertex $i$
5  $v_c \leftarrow v_0$ ;                                       // current vertex
6  $v_h \leftarrow v_0$ ;                                       // home vertex
7  $V_n \leftarrow N_G(v_c)$ ;                                  // all vertices adjacent to $v_c$
8  $V_d \leftarrow V_d \cup V_n$;
9  while $V_d \neq \emptyset$ do
10     $t_r \leftarrow$ Query_Deadline() ;                      // Check deadline
11     if $t_r$ is unknown then
12         $v_t \leftarrow$ NextVertexWODeadline($V_d, v_c$);
13     else if $t_r$ is 0 then
14         return $G$
15     else
16         $v_t \leftarrow$ NextVertexWDeadline($V_d, v_c, v_h, t_r$);
17     Move agent to $v_t$ through graph $G$;
18     $V_d \leftarrow V_d \setminus v_t$; $v_c \leftarrow v_t$; $V_{vis} \leftarrow V_{vis} \cup v_c$ ;
19     Add ($v_c$, $V_n$, E($v_c$)) to $G$;
20     $V_n \leftarrow N_G(v_c) \setminus V_{vis}$;               // exclude visited vertices
21     $V_d \leftarrow V_d \cup V_n$ ;                          // update discovered vertices
22 if $t_r$ is unknown then
23     Move agent to $v_h$ through $G$;
24 return $G$;
```

2.3.3 Stage III: Prioritized Exploration Algorithm

The input to the prioritized exploration algorithm (Algorithm 1) is the starting position v_0 of the robot, also called the home vertex v_h. The output is the explored graph G under the constraint that the robot should return to the home position by the end of exploration. The deadline as shown in the algorithm is queried from a function `Query_Deadline()`. The variable t_r is used to keep track of the shrinking deadline as the robot explores the environment. Note that the known initial value of t_r is t_{r_0}.

When the deadline is unknown, the exploration algorithm employs the function `NextVertexWODeadline` to calculate the next vertex, v_t, that the robot should visit

Algorithm 2: `NextVertexWODeadline`: Vertex selection without deadline

```
    Input: $V_d$, $v_c$
    Result: Next vertex to visit
1   $v_t \leftarrow$ initialize from $V_d$;
2   for $v_d$ in $V_d$ do
3       if $p(v_d) > p(v_t)$ then
4           $v_t \leftarrow v_d$;
5       if $p(v_d) = p(v_t)$ then
6           $cost_{cd} \leftarrow w_{cd}$; $cost_{ct} \leftarrow w_{ct}$;
7           if $cost_{cd} < cost_{ct}$ then $v_t \leftarrow v_d$;
8   return $v_t$;
```

without a deadline. Conversely, when the deadline is known, the algorithm utilizes the function `NextVertexWDeadline` for this calculation. Each of these functions returns a target vertex v_t for the robot to visit. Upon visiting the target vertex, v_t is added the set of visited vertices V_{vis} and subtracted from the set of discovered vertices V_d. Subsequently, the robot's view from v_t, i.e., the adjacent vertices of v_t, is added to the list of discovered vertices V_d. Both the vertices V_{vis} and V_d, along with the edges incident to these vertices, are added into the exploration graph G. At each iteration, the robot selects a vertex from V_d for its visit. The environment is considered to be complete explored when there are no more discovered vertices to visit, that is, when $|V_d| = 0$.

2.3.3.1 Exploration without a Deadline

When exploring without a deadline, the agent performs a greedy selection of the vertex with the highest priority from the set of discovered vertices V_d as shown in Function `NextVertexWODeadline` (see Algorithm 2). If there are multiple vertices with the highest priority, the agent chooses the one with the lowest cost. The cost is the estimated time to traverse the shortest path between the current vertex v_c and the candidate target vertex. If there are multiple such vertices which has the same priority and the same cost, a vertex is randomly chosen among equally good vertex options. The output of this function is the target vertex v_t the robot should head to.

2.3.3.2 Exploration with a Deadline

Algorithm 3: `NextVertexWDeadline`: Vertex selection with deadline

```
   Input: $V_d$, $v_c$, $v_h$, $t_r$
   Result: Vertex to visit next
1  $cost_{ch} \leftarrow w_{ch}$;
2  if $cost_{ch} < t_r$ then // Return home is possible
3      $V_{eligible} \leftarrow \emptyset$;
4      for $v_d$ in $V_d$ do
5          $cost_{cd} \leftarrow w_{cd}$; $cost_{dh} \leftarrow w_{dh}$;
6          if $(cost_{cd} + cost_{dh}) < t_r$ then
7              add $v_d$ to $V_{eligible}$;
8      if $V_{eligible}$ is $\emptyset$ then return $v_h$;
9      $v_t \leftarrow \arg\max_{v_d \in V_{eligible}} p(v_d)$, tiebreakers: $cost_{cd}, cost_{dh}$;
10     return $v_t$
11 else // Return home not possible
12     $V_{eligible} \leftarrow \emptyset$;
13     for $v_d$ in $V_d$ do
14         $cost_{cd} \leftarrow w_{cd}$;
15         if $cost_{cd} < t_r$ then
16             add $v_d$ to $V_{eligible}$;
17     if $V_{eligible}$ is $\emptyset$ then
18         return $v_c$
19     $v_t \leftarrow \arg\max_{v_d \in V_{eligible}} p(v_d)$, tiebreakers: $cost_{cd}, cost_{dh}$;
20     return $v_t$
```

When the deadline is known to the robot during exploration, it computes whether the deadline provides enough time for it to return to the home vertex. If the time limit t_r is adequate, the robot checks if the time limit allows it to explore additional vertices in V_d before returning to the home vertex v_h. It identifies a subset $V_{eligible}$ of V_d that consists of the vertices the robot would be able to visit and still return home within the deadline. The robot selects the vertex with the highest priority in this subset $V_{eligible}$. If there are several such vertices with equal priority, the robot chooses to visit the vertex with the lowest path costs $cost_{cd}$ and $cost_{dh}$; $cost_{cd}$ is the path cost to travel from the current vertex v_c to vertex $v_d \in V_{eligble}$, and $cost_{dh}$ is the

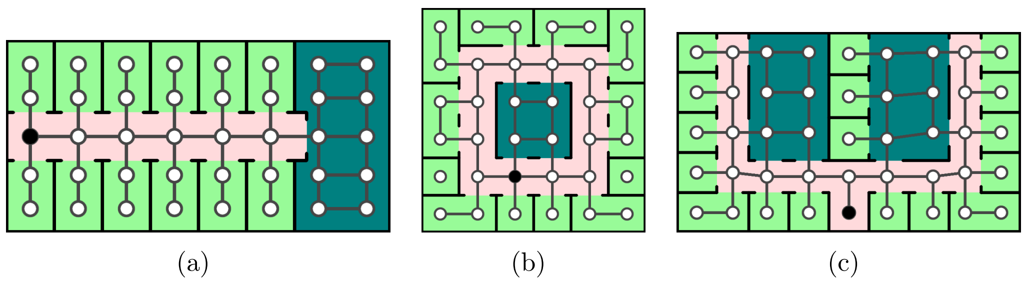


Figure 2.4: Three indoor graph environments showing different corridor layouts used in our exploration algorithm simulations. The respective graph environments are overlaid on them. Light green represents small rooms, dark green represents large rooms, and pink represents corridors. (a) Straight Corridor ends in a large room, with rooms on either side. (b) Looped Corridor wraps around a large room, connecting multiple rooms on its periphery. (c) Branched Corridor splits into two, connecting large and small rooms. The environments are shown to scale.

path cost to travel from v_d to the home vertex v_h. If the time is inadequate for the robot to reach the home vertex, the robot continues exploring the environment until the deadline so it can communicate the explored map back to its base. This behavior can be modified to instead make the robot return as close as possible to the home location. The $cost_{dh}$ is used as a tiebreaker when return is not possible, so the robot can come closer to the home location. See the pseudocode in Algorithm 3.

2.4 Experiments

2.4.1 Simulation Environment Setup

Typically indoor environments employ corridors to connect multiple rooms and other corridors in a building. A corridor is an architectural element that functionally leads to rooms, or that loops back to itself while connecting to rooms, or that branches off to several corridors, which in turn connect to rooms. Rooms can be categorized into large rooms which connect to further small rooms. Large rooms are usually atriums, halls, or galleries. Small rooms are usually small individual offices or closets. Usually, in a large indoor environment, we find a combination of all three building structures.

2.4.1.1 Graph Environments

These simulated environments, represented as graphs, are based on the three types of corridor layouts (Figure 2.4). Figure 2.4(a) shows an environment with a straight line corridor ending in a large room while connecting small rooms on either side. Figure 2.4(b) shows an environment where a corridor wraps around a large room while connecting to several rooms on its outer periphery. Figure 2.4(c) shows an environment with a corridor that branches into two separate corridors. Each of these separated corridors has large and small rooms connected to them. As large rooms can provide connectivity to other building structures, this layout has the large room on the right connecting to several smaller rooms.

2.4.1.2 Gazebo Environments

A set of three environments (Figure 2.5) similar to the graph environments has been created in Gazebo [24]. These environments are explored using a simulated Turtlebot3 robot equipped with a single scan 360° LiDAR with a range of 6 meters, matching the Slamtec RPLiDAR A1M8 sensor. The robot starts from a specified location. We use the GMapping SLAM algorithm [25] to create an occupancy grid map from the sensor data. This occupancy grid map is converted to G as discussed in Section 2.3.1. A new skeleton graph is created after the robot reaches the target vertex while exploring the map. This adds a few challenges: First, the graph needs to be recomputed every time the map is updated as some of the frontier vertices may cease to exist once the environment is explored further past the current frontier. Second, a list of visited vertices V_{vis} cannot be maintained as previously existing vertices may not exist later on during the exploration. As the robot follows the edges of the skeleton graph, the edges and vertices should be at a safe distance from the obstacles to avoid collisions. Third, the SLAM algorithm ignores LiDAR sensor values of infinity that are reported when there are no obstacles within the sensor range in a region. Such regions are not

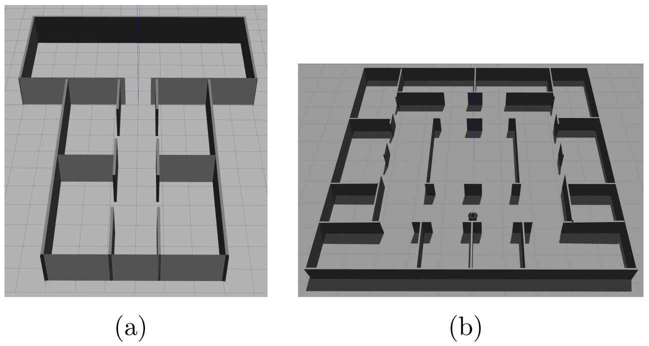
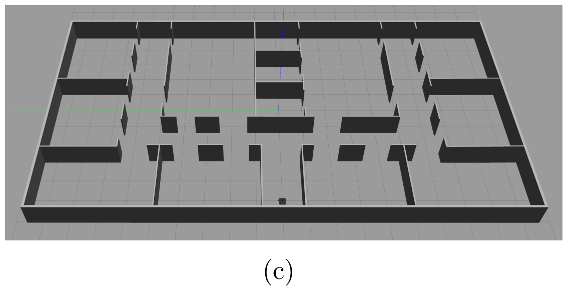
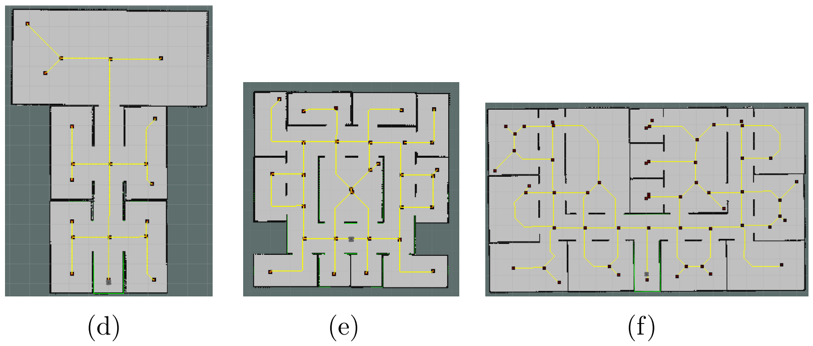




Figure 2.5: Three 3D Gazebo environments with different corridor layouts used in simulating our exploration algorithm. (a) Straight Corridor Gazebo environment in 3D, (b) Looped Corridor Gazebo environment, (c) Branched Corridor Gazebo enironment. (d, e, f) Occupancy grid maps of the environments, where yellow lines represent edges and black points represent vertices. (d) Straight Corridor environment. (e) Looped Corridor environment. (f) Branched Corridor environment.

mapped.

We address these challenges with the following steps: We update the graph once the robot reaches the target vertex, instead of every time the map is updated by the SLAM algorithm. We maintain the locations that the robot has visited as a substitute for the set of visited vertices. If there are new vertices that are created within a threshold distance from the stored locations, we ensure that the vertices are not a part of V_d. To ensure collision-free edges between vertices, we make a binary map of the occupancy grid, with the unoccupied space marked with a different value than the rest of the map. This unoccupied space is eroded by the robot's width. This eroded map is then converted to a skeleton graph. This ensures that the edges of the skeleton graph are at a safe distance from the obstacles.

2.4.2 Real-world experiments

Real-world experiments introduce a set of additional challenges. Simultaneous Localization and Mapping (SLAM) is more challenging as robot motion introduces errors that alter the map while the robot explores the environment. During experiments, we observed that the robots could not successfully navigate to goal positions by following the edges of the exploration graph, as the map underwent significant updates during the robot's movement. Additionally, reflective surfaces such as glass, varnished wooden doors, and reflective metal surfaces contribute to mapping inaccuracies. The mobile robot's wheel design was also critical in the choice of deciding the mobile robot to use. We found that the Turtlebot3 Waffle Pi robot was highly susceptible to wheel odometry noise due to its smaller and slippery wheels.

To accomodate these practical challenges, we use the `move_base` package from the ROS Navigation stack to navigate the robot from the current to the goal position. The move base package utilizes the Dynamic Window Approach (DWA) [26] to calculate a trajectory from the current position to the goal position. We found the ROS Navigation tuning parameters from [27] to be very helpful to tune the `move_base` parameters. Due to the wheel slipping of the Turtlebot3 Waffle Pi, we used the AgileX Limo Robot.

The robot (see Figure 2.6) was connected to the laptop computer using an existing Wi-Fi network with several access points. While the robot maintained an active connection with the laptop during the exploration process, the connection bandwidth would drop as the robot moved from one place to another. In practice, using a direct Wi-Fi connection between the robot and the laptop would generate a connection with

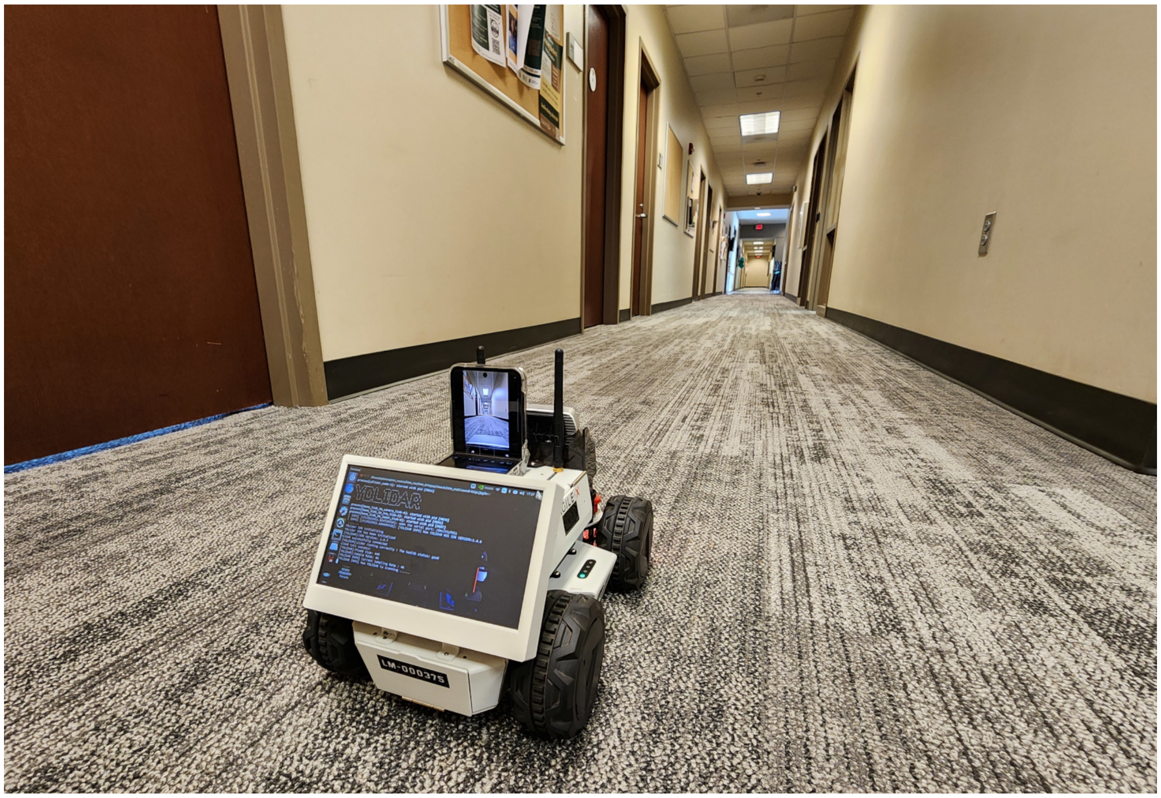


Figure 2.6: AgileX LIMO robot with camera mounted on top.

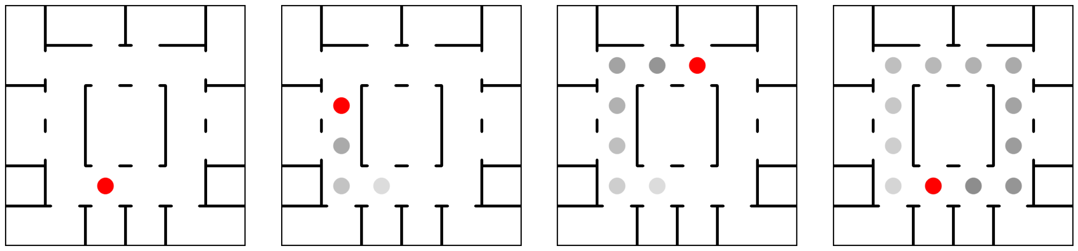
higher reliability.

2.4.3 Metrics

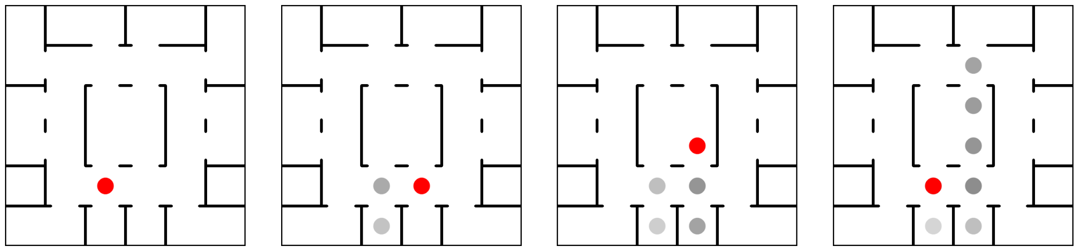
The performance of the exploration algorithms is measured using the extent of the explored environment. We compare our Priority-based greedy exploration algorithm to the Cost-based greedy exploration algorithm. The Cost-based exploration algorithm is motivated by a baseline greedy exploration algorithm [1]. As the agents are informed of the deadline at time instant t_a, we compare the performance of the agents at time $t_a + t_{r_0}$.

For the graph environments, explored regions of the map are represented by the set of explored vertices V, where $V = (V_d \cup V_{vis})$. The performance metric for the graph environments is $|V|$, the number of explored vertices.

Since the number of vertices may change during exploration for the Gazebo environments, the performance metric used is the percentage of floor area explored. Here



(a) Exploration by Priority-based exploration algorithm.



(b) Exploration by Cost-based exploration algorithm.

Figure 2.7: Exploration of the Looped Corridor graph environment. Each row shows snapshots of exploration at time steps 0, 6, 12, and 25. The deadline t_{r_0} of 10 time steps was provided at a t_a of 15 time steps. Each edge cost is 2 time steps. The red circle denotes the current position of the robot and the trail of circles show the visited vertices. (a) Priority based exploration strategy, which prioritizes corridors over large rooms and small rooms. (b) Cost-based exploration strategy, which gives equal priority to all vertices.

the exploration time includes the computation time, and the times t_a and t_{r_0} are measured in seconds.

2.5 Results

2.5.1 Graph Environments

The Priority-based exploration algorithm and Cost-based exploration algorithm are evaluated on three graph environments (Figure 2.4) in Figures 2.8, 2.9, and 2.10. We present a few detailed results in Tables 2.1, 2.2, and 2.3. For these experiments, we provided the agents with the deadline t_{r_0} at the start of exploration. So t_a is zero. The deadline t_{r_0} has been set to three different values: a large deadline, an intermediate deadline, and a short deadline represented by the values 500, 60, and 30 respectively. Since Cost-based Greedy(C-Greedy) randomly chooses between two vertices of the same cost, the values in the tables are averaged over 100 simulations for each deadline.

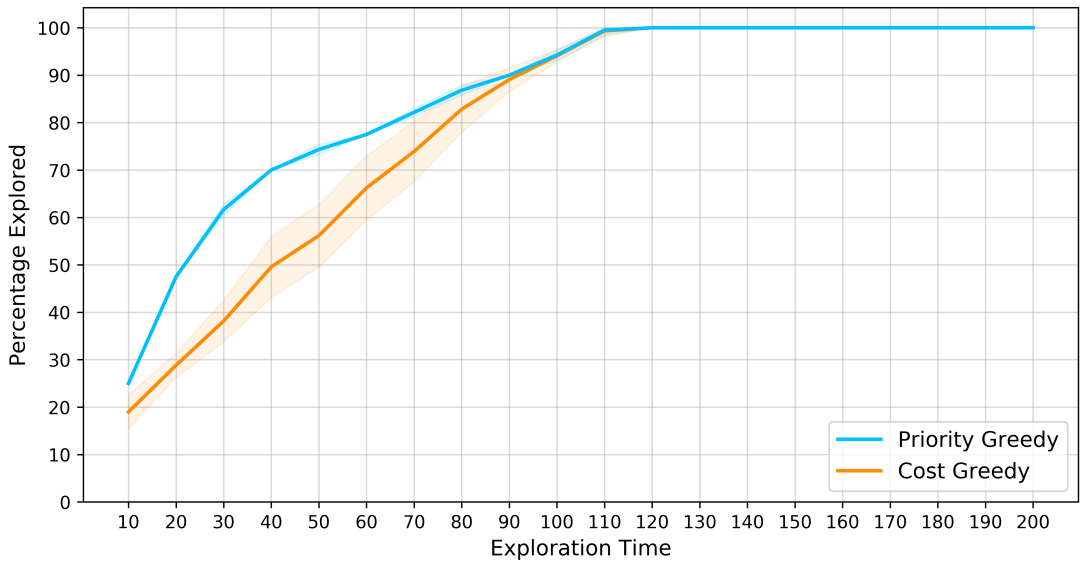


Figure 2.8: Comparison between the Priority-based Greedy and the Cost-based Greedy algorithm in the Straight Corridor graph environment. Each value on the x-axis shows an independent experiment with a specified exploration time. The y-axis value shows the percentage of the total environment explored. The curve shows the mean performance of each algorithm. The shaded region shows the standard deviation of thirty independent experiments.

A comparison of exploration by Priorty-based greedy exploration algorithm (P-Greedy) and Cost-based greedy exploration algorithm (C-Greedy) in the looped corridor graph environment is shown in Figure 2.7. It shows a trail of explored vertices as an agent explores the environment of Figure 2.4(b).

For all three environments, in the case of a large deadline $t_{r_0} = 250$, we observe that both agents have explored the entire graph environment. For an intermediate deadline of $t_{r_0} = 60$, P-Greedy explores a higher percentage of all vertices than C-Greedy. Since P-Greedy prioritizes corridors over other regions, its corridor exploration percentage is higher than for C-Greedy in all the environments. For the Straight Corridor environment, C-Greedy explores a higher percentage of small rooms than P-Greedy while P-Greedy explores a higher percentage of the large room than C-Greedy. For the short deadline of $t_{r_0} = 30$, P-Greedy explores a higher portion of the environment than C-Greedy for all three graph environments. As small rooms and large rooms are adjacent to the corridor, P-Greedy eventually explores a higher percentage of these

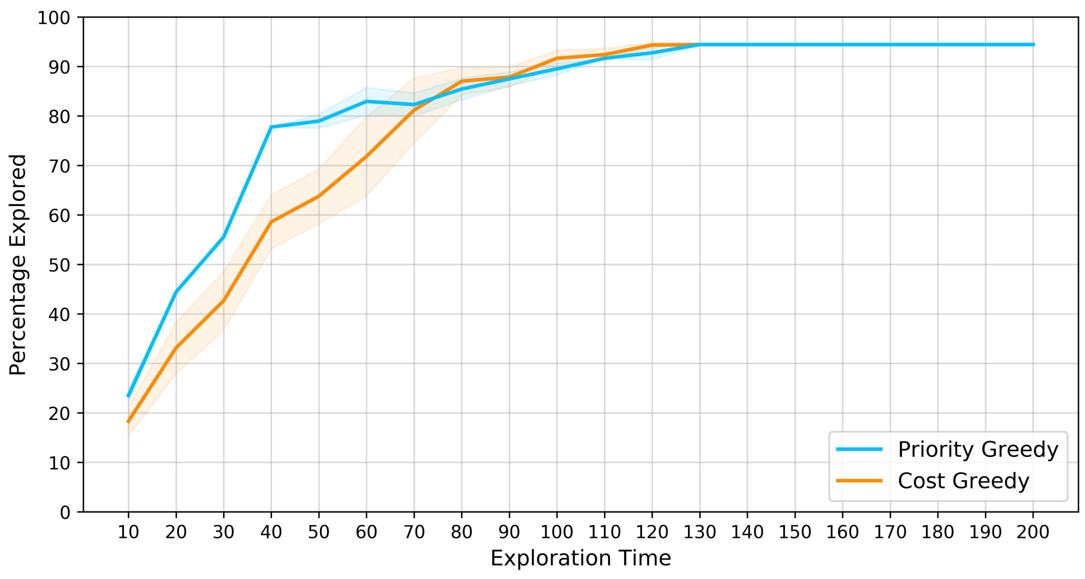


Figure 2.9: Comparison between the Priority-based Greedy and the Cost-based Greedy algorithm in the Looped Corridor graph environment.

Table 2.1: Exploration results for the graph straight corridor environment.

t_{r_0}	**Agent**	**Corridor**	**Large Room**	**Small Room**	**All Vertices**
250	P-Greedy	100.0%	100.0%	100.0%	100.0%
	C-Greedy	100.0%	100.0%	100.0%	100.0%
60	P-Greedy	100.0%	**100.0%**	61.4%	**76.8%**
	C-Greedy	99.3%	22.6%	**73.0%**	64.4%
30	P-Greedy	**100.0%**	**66.8%**	**50.0%**	**61.7%**
	C-Greedy	75.2%	0.3%	43.5%	37.4%

sections while visiting the corridor vertices.

To summarize, P-Greedy's prioritization of corridor vertices has allowed it to discover more vertices, thereby increasing the overall exploration percentage.

2.5.2 Gazebo Environments

Exploration algorithms P-Greedy and C-Greedy were evaluated, based on the percentage of explored floor area, on the three Gazebo environments shown in Figure 2.5. The deadline t_{r_0} to return to the home location is provided to the robot after t_a seconds. If the deadline is provided at the start, $t_a = 0$, else $t_a > 0$.

For our experiments, we have three sets of values for t_a and t_{r_0}. The values $t_a = 0$

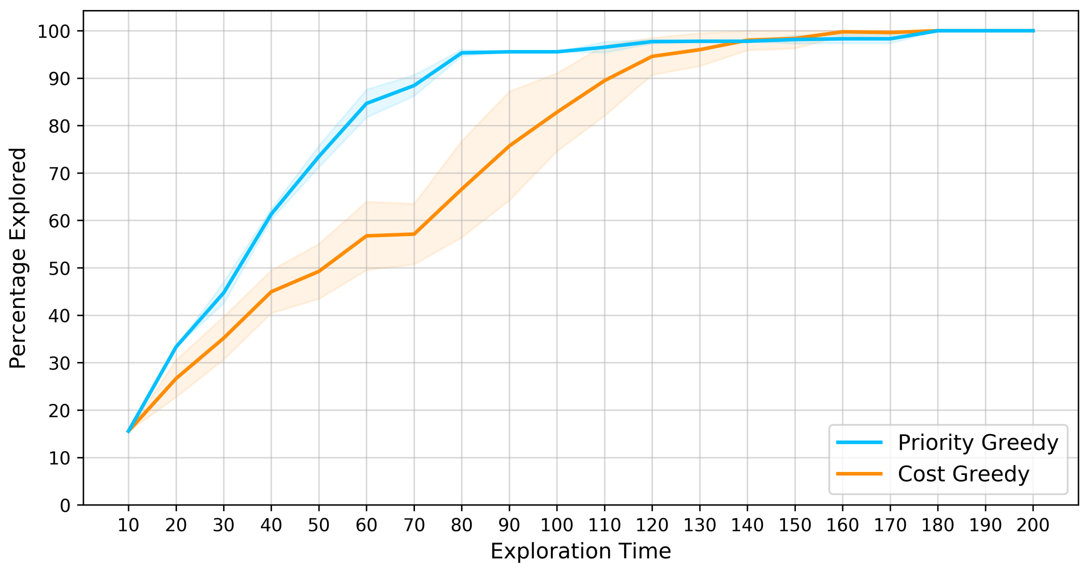


Figure 2.10: Comparison between the Priority-based Greedy and the Cost-based Greedy algorithm in the Branched Corridor graph environment.

Table 2.2: Exploration results for the graph looped corridor environment.

t_{r_0}	Agent	Corridor	Large Room	Small Room	All Vertices
250	P-Greedy	100.0%	100.0%	90.0%	94.4%
	C-Greedy	100.0%	100.0%	90.0%	94.4%
60	P-Greedy	**100.0%**	100.0%	**65.0%**	**80.6%**
	C-Greedy	93.8%	99.0%	59.3%	75.2%
30	P-Greedy	**83.3%**	**85.8%**	**40.0%**	**59.5%**
	C-Greedy	63.1%	71.5%	28.5%	44.8%

and $t_{r_0} = 500$ signify a short exploration time provided at the start of the exploration. The other two cases $t_a = 500$, $t_{r_0} = 500$ and $t_a = 500$, $t_{r_0} = 1000$ correspond to intermediate and long exploration times. The exploration results for the three environments, averaged over three simulations for each exploration time, are presented in Tables 2.4, 2.5, and 2.6.

For the straight corridor environment (Figure 2.5(b)), with the longest exploration time, $t_a = 500$, $t_{r_0} = 1000$, both P-Greedy and C-Greedy almost completely explore the environment. P-Greedy explores a higher percentage of the large room that lies at the end of the corridor while C-Greedy explores a higher percentage of small

Table 2.3: Exploration results for the graph branched corridor environment.

t_{r_0}	Agent	Corridor	Large Room	Small Room	All Vertices
250	P-Greedy	100.0%	100.0%	100.0%	100.0%
	C-Greedy	100.0%	100.0%	100.0%	100.0%
60	P-Greedy	**100.0%**	**75.0%**	**75.0%**	**82.8%**
	C-Greedy	72.6%	62.1%	46.6%	58.8%
30	P-Greedy	**64.3%**	32.3%	**36.8%**	**44.2%**
	C-Greedy	51.6%	32.1%	21.8%	33.8%

Table 2.4: Exploration in the Gazebo straight corridor environment

t_a	t_{r_0}	Agent	Corridor	Large Room	Small Room	Total
500	1000	P-Greedy	99.8%	**98.7%**	96.7%	**98.1%**
		C-Greedy	99.6%	77.2%	98.6%	88.7%
500	500	P-Greedy	100.0%	**24.1%**	96.6%	**63.0%**
		C-Greedy	97.7%	6.1%	96.5%	54.3%
0	500	P-Greedy	85.3%	0.0%	65.8%	37.7%
		C-Greedy	84.7%	0.0%	62.5%	36.3%

rooms. For the intermediate exploration time, $t_a = 500$, $t_{r_0} = 500$, P-Greedy explores a higher percentage of the large room, while C-Greedy explores the smaller rooms before returning to the home location. On average, P-Greedy has explored 24% of the large room, while C-Greedy has explored 6% of the large room. When the exploration time is short, $t_a = 0$, $t_{r_0} = 500$, P-Greedy explores 38% of the total environment while C-Greedy explores 36% of the total environment. For this case, for every graph update, the nearest frontier vertex to the robot is coincidentally the vertex with the highest priority. So P-Greedy and C-Greedy visit almost the same set of locations during exploration. That the area of the corridor explored by both P-Greedy and C-Greedy is equal (85% of the corridors) quantifies this observation.

The looped corridor environment of Figure 2.5(c) is much smaller than the other two environments. For exploration with the longest exploration time, $t_a = 500$, $t_{r_0} = 1000$, both P-Greedy and C-Greedy have explored the entire environment.

Table 2.5: Exploration in the Gazebo looped corridor environment

t_a	t_{r_0}	**Agent**	**Corridor**	**Large Room**	**Small Room**	**Total**
500	1000	P-Greedy	100.0%	99.7%	97.6%	98.7%
		C-Greedy	100.0%	99.3%	97.8%	98.9%
500	500	P-Greedy	100.0%	99.2%	96.1%	97.9%
		C-Greedy	100.0%	99.6%	96.8%	98.3%
0	500	P-Greedy	**89.0%**	99.6%	**76.0%**	**83.5%**
		C-Greedy	85.6%	99.5%	57.7%	72.6%

With an intermediate exploration time of $t_a = 500$, $t_{r_0} = 500$, both exploration agents explore almost all of the unknown environment. When the exploration time is short, $t_a = 0$, $t_{r_0} = 500$, P-Greedy explores more of the corridor and of the total environment (83.5%) compared to C-Greedy's total exploration of 72.6%.

Table 2.6: Exploration in the Gazebo branched corridor environment

t_a	t_{r_0}	**Agent**	**Corridor**	**Large Room**	**Small Room**	**Total**
500	1000	P-Greedy	**80.3%**	**66.7%**	**64.8%**	**68.6%**
		C-Greedy	77.5%	54.7%	62.6%	61.4%
500	500	P-Greedy	**68.2%**	54.6%	**48.5%**	**54.3%**
		C-Greedy	55.6%	54.8%	21.5%	36.8%
0	500	P-Greedy	**50.4%**	19.7%	**21.5%**	**27.4%**
		C-Greedy	43.3%	18.7%	14.2%	21.6%

The branched corridor environment of Figure 2.5(d) is much larger than the other two environments. Hence, both P-Greedy and C-Greedy could not explore the entire environment even with the longest exploration time, $t_a = 500$, $t_{r_0} = 1000$. The performance difference between P-Greedy and C-Greedy is the lowest for this case because the cost of returning back home limits the exploration in the second branch for both agents. When $t_a = 500$, $t_{r_0} = 500$, the difference is more pronounced as P-Greedy explores one complete branch of the corridor and a part of the other corridor, while C-Greedy explores significantly less. However, for the shortest exploration time, $t_a = 0$, $t_{r_0} = 500$, P-Greedy completely explores one of the two branches of the corridor, while C-Greedy's bias towards the closest vertex limits the extent of its

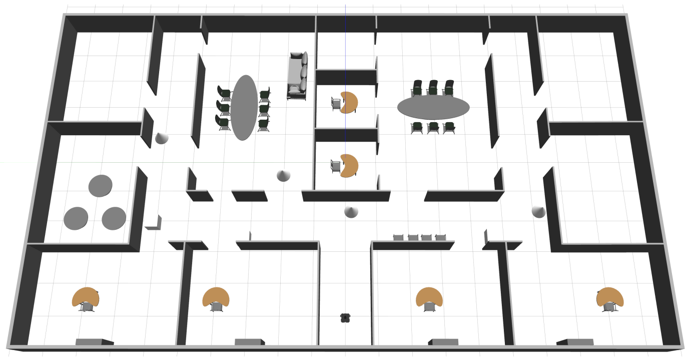


Figure 2.11: Branched Corridor with obstacles. A Gazebo environment showing a Branched Corridor exploration environment with several obstacles.

exploration.

Results on Gazebo environment with obstacles: Our Gazebo experiments were initially conducted in environments devoid of obstacles typically found in indoor structures, such as office buildings. To address this limitation, we created a modified version of the branched corridor environment, incorporating obstacles like furniture and other static objects, as depicted in Figure 2.11. We name this environment, "Branched Corridor with obstacles". Given that our priority allocation relies on a geometry-based method, the presence of these obstacles introduced occlusions in the environment, thereby posing a challenging classification problem. In this scenario, the classifications were incorrect several times, especially when a obstacles would throw off the estimate of the room size.

Table 2.7: Exploration result in the Occluded Branched Corridor

t_a	t_{r_0}	Agent	Corridor	Large Room	Small Room	Total
500	1000	P-Greedy	**80.0%**	**69.1%**	44.1%	57.8%
		C-Greedy	72.9%	63.2%	41.7%	53.6%
500	500	P-Greedy	71.9%	60.0%	**30.6%**	46.5%
		C-Greedy	72.2%	57.6%	25.7%	43.4%
0	500	P-Greedy	**57.9%**	35.4%	**12.9%**	**27.9%**
		C-Greedy	52.2%	33.7%	2.3%	20.5%

Table 2.7 presents the results in the Gazebo environment, Branched Corridor with Obstacles. In this environment, we observe that P-Greedy consistently outperforms C-Greedy across all three deadlines. However, the presence of obstacles results in a reduced overall explored area for both P-Greedy and C-Greedy compared to the Branched Corridor environment without obstacles. During these experiments, we noted that both P-Greedy and C-Greedy occasionally collided with the obstacles. This occurred because the LiDAR sensor scans the environment at a height slightly above that of the obstacles, leading it to report these areas as free in the occupancy grid map. Employing a different LiDAR sensor with multiple scan lines, such as the Velodyne VLP-16, or a 3D depth camera like the Microsoft Kinect or Intel RealSense stereo camera, could mitigate this issue.

We observe that the explored percentage for P-Greedy is consistently higher than or equal to that for C-Greedy. By visiting the corridors first, P-Greedy often explored the neighboring regions of small and large rooms. The performance on the Gazebo environments is not as high as on the graph environments due to the additional challenges discussed earlier. We observed that the robots spend time to travel back to the skeleton graph, instead of traveling directly to a target vertex. Even though this impacts the exploration performance, it ensures collision-free paths. Additionally, changes in the locations of obstacle boundaries in the occupancy map from the SLAM algorithm affect the accuracy of priority classification of the environment regions.

2.5.3 Real-world Experiments

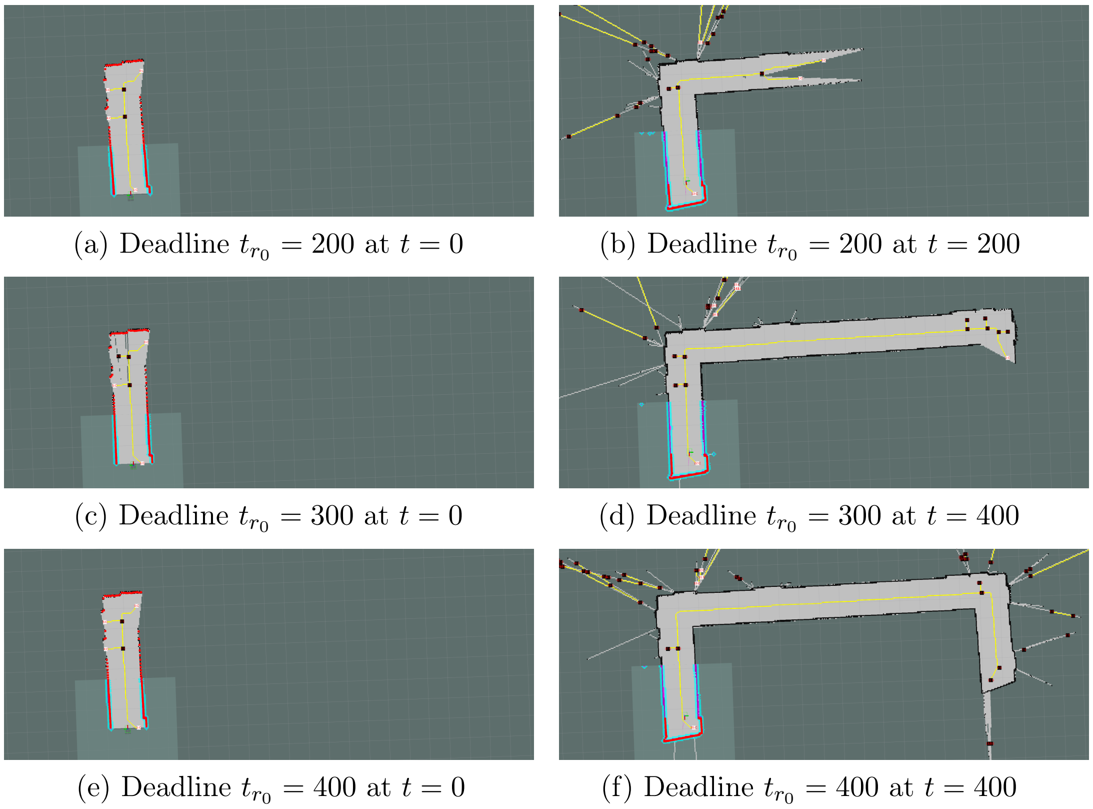
In the real-world experiments conducted with the AgileX LIMO robot, we set three exploration deadlines: 200, 300, and 400 seconds. These time constraints were imposed at the start of each exploration experiment. During these tests, the robot consistently managed to return to its starting location within the deadline. The results for these three deadlines are illustrated in Figure 2.12.

The robot utilized the Priority-based Greedy algorithm to explore the environment. We conducted two independent trials for each deadline, with the algorithm demonstrating consistent performance. Notably, the computation times were longer in the real-world experiments compared to simulations. This increase was partly due to the infrequent map updates, occurring every two seconds despite the GMapping SLAM [25] setting, which was configured to update the map at 5Hz. The slower computation performance of the robot likely contributed to this delay. The limited compute capability on the robot led us to divide computation between the robot and a laptop connected on the WiFi network. We ran the SLAM algorithm and DWA Planner [26] on the robot and the path planning algorithm on a laptop, thereby introducing further delays due to network latency. The physical maps generated during these experiments were also notably noisier than those in simulated Gazebo environments, likely due to surface reflections from materials like metal, glass, and varnished wooden doors.

The speed and extent of robot exploration was noticeably reduced in the real-world scenarios compared to the Gazebo simulations, largely because of the increased computation time as the robot awaited map updates. Challenges in localization and mapping accuracy were particularly pronounced in featureless corridors, where the lack of distinct landmarks significantly limited the SLAM algorithm's capability to accurately map the environment. To enhance the results, we could consider several improvements: employing a more diverse array of sensors beyond the single-scan LiDAR, performing computations directly on the robot's onboard computer to reduce network delays, and adopting a SLAM algorithm better suited for fast-moving robots in indoor settings. These adjustments could potentially mitigate the issues observed and improve the efficiency and reliability of the robotic exploration.

2.6 Contributions

In this chapter, we have developed a technique using skeleton graphs to build an exploration graph from the occupancy grid map created from the SLAM algorithm.


(a) Deadline $t_{r_0} = 200$ at $t = 0$ (b) Deadline $t_{r_0} = 200$ at $t = 200$

(c) Deadline $t_{r_0} = 300$ at $t = 0$ (d) Deadline $t_{r_0} = 300$ at $t = 400$

(e) Deadline $t_{r_0} = 400$ at $t = 0$ (f) Deadline $t_{r_0} = 400$ at $t = 400$

Figure 2.12: Exploration results on the AgileX LIMO robot using the Priority-based greedy algorithm with different deadlines. Each row shows a different experiment, where a deadline has been imposed on the robot at the start of the exploration. The darker gray cells denote the unmapped area, the lighter gray cells denote the mapped and obstacle-free area, and the black cells denote the obstacles. The red cells are LiDAR scans. The robot explores larger areas of the map with larger deadlines. Each time, the robot comes back to the starting position by the deadline.

This approach enables us to use graph-based exploration algorithms for addressing the prioritized robot exploration problem. Subsequently, we developed a short-horizon Priority-based Greedy exploration algorithm. This algorithm is designed to explore the initially unknown environment and ensure the robot returns to the home vertex within a dynamically imposed deadline.

2.6.1 Future Work

There are several directions for improvement of the graph exploration algorithm. In our experiments, building structures were classified and assigned user-specified vertex priorities based on local occupancy grid maps. Moving beyond a geometry-based

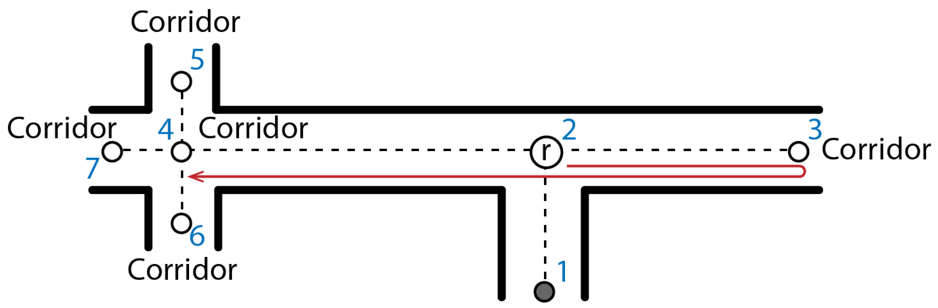


Figure 2.13: This figure shows the limitation of the greedy algorithm. The position of the robot is shown by the vertex marked ‘r’. The vertices are circles, the white circles are discovered vertices yet to be visited by the exploration algorithm and gray circles are visited vertices. All the discovered vertices are classified as ‘Corridor’ vertices. The greedy algorithm chooses to visit the closest corridor vertex (Vertex 3) followed by the Vertex 4. However, as more corridor vertices (5,6, and 7) are present on the left, the robot should have visited Vertex 4 first for an optimal route. This shows the need of algorithms better than the greedy algorithm.

classification, we could employ computer vision techniques and neural network-based classifications to achieve a more robust semantic classification of regions. Additionally, it would be beneficial to dynamically adjust priority values based on the connectivity of each region.

The Priority-based Greedy algorithm as shown in this chapter is a short-horizon exploration algorithm as it plans with a single step lookahead, which may not produce an optimal solution. For example, in Figure 2.13, we observe that the Priority-based Greedy algorithm makes the robot choose a suboptimal choice guiding the robot to a single corridor vertex on the right (Vertex 3) instead of the cluster of corridor vertices (Vertices 4, 5, 6, and 8) to the left. This limitation can be mitigated by developing a long-horizon exploration algorithm. Such algorithms can consider the sequence of vertices visited by the robot to maximize the number of vertices visited while considering the priority of the vertices. To address these challenges, Chapter 4 explores mixed integer linear programming approaches for the prioritized exploration problem with dynamic deadlines.

2.7 Conclusion

In this chapter, we introduced a Priority-based greedy exploration algorithm tailored for indoor environments, subject to the constraint of dynamic deadlines. The objective was to rapidly determine the geometric structure and connectivity of the environment. The exploration environment was represented as a graph, with the map and robot position serving as inputs to the algorithm. Our algorithm prescribes a prioritized order for exploring discovered vertices to maximize the exploration of the graph. Additionally, it ensures that, given adequate time, the robot can return to its home location upon completion of the exploration. We have evaluated our algorithm in simulation environments in Gazebo and real-world experiments to demonstrate its significant improvement over non-prioritized exploration approaches.

Our findings indicate a marked improvement in performance, particularly under short deadlines, both in the graph-based environments and the Gazebo simulations, when employing the Priority-based Greedy exploration algorithm.

CHAPTER 3: MILP FORMULATION FOR PRIORITIZED INDOOR ROBOT EXPLORATION WITH DYNAMIC DEADLINES

3.1 Introduction

The prioritized exploration problem, as introduced in Chapter 2, is a variant of the exploration problem where the robot must explore an initially unknown environment to compute its layout (i.e., connectivity) maximally while returning to the home location within a deadline. In Chapter 2, we introduced a priority-based greedy solution that attempts to solve the prioritized exploration problem by creating a path that visits high priority regions while ensuring the robot returns to the home location within the deadline, with an assumption that high priority regions provide better views of the unknown environment.

To explore the environment quickly and efficiently, the robot should ideally visit each target location at most once. In practice, with limited knowledge of the environment, the robot should minimize revisiting locations that it has already visited. This connects our problem to the Hamiltonian Path Problem and node routing problems such as the Traveling Salesperson Problem (TSP). These well known problems have been used to model coverage problems where each vertex needs to be visited only once [28, 29]. This chapter considers the following question: Would formulating the prioritized exploration problem as the Orienteering Problem (OP), Profitable Tour Problem (PTP), or Minimum Latency Paths (MLP) problem, or Profitable Tour Problem with Minimum Latency Paths (PTP-MLP) improve exploration performance?

The OP, PTP, MLP problem, and PTP-MLP are NP-complete problems [30] and require time exponential in the number of vertices to solve optimally. In this chapter, when needed we reduce the number of candidate vertices to a feasible size so the prioritized exploration problem can be solved rapidly.

The chapter discusses the relevant literature in Section 3.2, and describes adaptation of the Orienteering Problem, Profitable Tour Problem, Minimum Latency Path Problem and Profitable Tour Problem with Minimum Latency Paths for the prioritized exploration problem in Section 3.4. Results of simulation experiments are presented in Section 3.5, followed by a discussion of the behavior of the exploration algorithms in Section 3.7.

3.2 Related Work

The problem of robotic exploration has been studied from multiple perspectives. The work described in this chapter builds on previous work on robot exploration, mapping techniques, and path planning.

Cost-based frontier exploration techniques that have been used for single-robot exploration [1] and multi-robot exploration [5] have laid the groundwork for robotic exploration. Next-best-view [31] approaches such as receding horizon next-best-view [32] have also been used for 3D mapping of environments. Robotic exploration has been modeled as different problems such as target searching, mapping, and coverage in unknown or partially observed environments [33, 34, 35, 36, 29].

Another approach for robot exploration is Active SLAM, a variant of the Simultaneous Localization and Mapping problem (SLAM). Active SLAM is the task of actively planning robot paths while simultaneously building a map and localizing within it [37, 36, 38]. Such techniques generate a consistent map that is essential for 3D inspection tasks, sometimes at the cost of slower exploration.

The task of robot exploration can be viewed as a variant of the Traveling Salesperson Problem (TSP) as the robot must compute a Hamiltonian path to visit the possible frontier locations to explore the map. Similarly, the Team Orienteering Problem and Orienteering Problem with Neighborhoods, variants of the prize-collecting

TSP with budget constraints, have been used to address the multi-robot exploration problem [39, 40].

Prior work has focused on exploration with goals such as finding a specific target, or consistent mapping at the cost of slower exploration. Most approaches consider the exploration to be complete when either the entire environment is mapped or the target is found in the unknown location. The prioritized exploration problem that we previously addressed [41] considers partial exploration of the environment, within a changing deadline. Single-robot exploration exploration using a mix of greedy and orienteering problem formulation has been addressed by **??**. A similar exploration problem is addressed by using a Multi-Robot Team Orienteering formulation in [39], where the environment is explored using heterogeneous robots with different energy limits. We are not aware of a single-robot exploration approach that addresses the prioritized exploration problem using the Profitable Tour Problem formulations.

3.3 Background

3.3.1 Orienteering Problem

The Orienteering Problem (OP) [42, 43, 44] belongs to the class of Traveling Salesperson Problems with profits, where it is not necessary to visit all vertices. The orienteering problem, derived from the sport of orienteering as discussed by Chao et al. [42], involves navigating through a natural terrain with a map and compass. Competitors aim to maximize their score by visiting as many control points as possible within a time constraint, requiring strategic selection of a subset of these points. The winner is determined by the highest total score achieved.

The input to the OP is a complete graph, a time budget, a start and an end vertex. Each graph vertex has an associated positive prize p_i. The goal of OP is to determine a path from the start to end vertex within the time budget, while selecting vertices to visit to maximize the total collected prize. The travel time between any two vertices is assumed to be non-zero. OP is suited to address problems where the time budget is limited compared to the time required to visit all vertices. When the time budget exceeds the time to visit all the vertices, the path generated by solving the OP formulation might not provide the shortest path to visit the vertices.

The Profitable Tour Problem also belongs to the class of TSPs with profits. To address the cost of the path, the Profitable Tour Problem (PTP) [45, 46] seeks to maximize the net profit, the total prize collected along the path minus the path cost. This formulation provides the shortest path when the time budget exceeds the time required to visit all the vertices.

The Minimum Latency Path(MLP) Problem [47] is a combinatorial optimization problem that aims to visit all vertices in a graph and minimizes the delivery time to each vertex. MLP Problem asks for a sequence to visit each of the vertices in a graph so that the total delivery time to all customers is minimized. This is different from Traveling Salesperson Problem (TSP) where the goal is to minimize the route cost of the salesperson. MLP Problem has been typically used in operations research, for customer centric routing problems or emergency logistics [48, 49, 50].

The Profitable Tour Problem with Minimum Latency Paths (PTP-MLP) is novel formulation and is a variant of the Profitable Tour Problem that maximizes the net profit, the total discounted prize collected along the path minus the cumulative delivery cost to vertices it visited. Here we consider a discounted value of the prizes of each vertex depending on the position of the vertex in the sequence of vertices the robot visits. Such discounting encourages the robot to visit high-prize vertices towards the start of the path.

The next section discusses modeling the prioritized robot exploration problem using the orienteering problem, profitable tour problem, minimum latency path problem, and profitable tour problem with minimum latency path formulations.

3.4 Problem Formulation

A single robot explores an unknown environment, senses and maps the area, and models the graph as an exploration graph $G = (V, E)$. We model this graph G as a complete graph of N vertices. Here, each vertex $i \in V$ has a prize p_i associated with it that represents its priority. The set of vertices V is divided into two subsets: V_d, the set of discovered vertices, and V_{vis}, the set of vertices visited by the robot. The discovered vertices are the set of vertices near the frontier of the explored region that has not been visited by the robot. E is the set of edges in G, where each edge $(i, j) \in E$ represents a collision-free path between i and j, and has a time cost t_{ij}. Both the prizes p_i and the time t_{ij} are non-negative values. A limited time budget of t_{max} is available to visit the vertices.

We hope to improve on the greedy method shown in Chapter 2 by introducing an orienteering problem formulation to the exploration problem. The exploration procedure may have a more efficient solution than the greedy formulation, thereby allowing the robot to explore a larger portion of the environment before returning to the home location. The objective of the exploration is to maximize the explored area and identify the connectivity of the indoor environment. It does so by creating an optimal path that visits high priority regions while ensuring the robot returns to the starting location within the deadline.

The orienteering problem creates an optimal path consisting of several discovered vertices. The robot marks the first vertex in the list of vertices in the path as the target vertex v_t and visits it. The target vertex v_t is marked as a visited vertex. Upon visiting the target vertex v_t, the robot can map a portion of the previously unexplored environment. The exploration graph is recomputed and the robot calculates the next sequence of vertices to visit. The formulation of the orienteering problem is introduced in Section 3.4.1.

3.4.1 Orienteering Problem Formulation

The objective of the orienteering problem [44, 43] is to generate a path that maximizes the total prize collected along it on the exploration graph. A path is defined as the sequence of vertices that the robot takes from the start to the end vertex. If the start and end vertex is the same vertex, then the path is called a tour. The prize at each vertex p_i is based on the priority of the vertex. For our formulation, we have set the corridor vertices to have the highest prize, followed by large rooms and finally small rooms. We do so in an attempt to relate the prize to the connectivity of each building structure. Corridors being highly connected portions of the environment, should provide us with the greatest information about connectivity and large rooms can be connected to smaller portions of the environment and provide lower information about connectivity. Small rooms are usually connected to fewer parts of the environment, providing very low connectivity. To adapt the orienteering problem to the exploration problem, the robot considers visiting a subset of the discovered vertices. The first discovered vertex the robot visits in the computed path from the orienteering problem is the target vertex. Once the robot visits a target vertex, a new path is computed accommodating the newly discovered vertices from the previously unexplored region. The binary variable $y_i \in \{0, 1\}$ determines if a particular vertex i is visited. The objective function is shown in Equation 3.1, where s is the start vertex and h is the home vertex. The binary variable y_i is 1 if vertex i is visited and 0 otherwise.

$$\text{Maximize} \sum_{\substack{i=1 \\ i \neq s,h}}^{N} p_i \, y_i \tag{3.1}$$

$$y_i \in \{0, 1\}, \; p_i \geq 0$$

The exploration graph is represented as a weighted adjacency matrix M. As the orienteering problem formulation requires a fully connected graph, an all-pairs shortest path algorithm [51] is used to create a fully connected distance matrix M_c. As the robot needs to visit only the discovered vertices, we create distance matrix M_{cd}, from M_c, to include only s, h and the discovered vertices. The number of vertices in M_{cd} is N. At the start of exploration, the robot starts exploring the initially unknown environment from the home vertex h, then we use the orienteering problem formulation to create a tour that starts and ends at vertex h. After the first step of exploration, when the robot is not at vertex h, the current vertex of the robot serves as the starting vertex s of the formulation and the end vertex is the home vertex h.

The robot's exploration is constrained by the total time constraint expressed in Equation 3.2, where $c_{i.j}$ is the time estimate to traverse edge $e_{i,j}$ from vertex i to vertex j. The binary variable $x_{i,j} \in \{0,1\}$ is 1 when the path contains the edge $e_{i,j}$ and 0 otherwise.

$$\sum_{\substack{i=1 \\ i \neq h}}^{N} \sum_{\substack{j=1 \\ j \neq i,s}}^{N} c_{i,j}\, x_{i,j} \quad \leq \quad t_r \tag{3.2}$$

$$x_{i,j} \in \{0,1\},\ t_r \geq 0,\ c_{i,j} > 0$$

Equations 3.3.1 – 3.3.3 present the set of inbound constraints. The robot is expected to return to the home vertex h at the end of the exploration. Hence the inbound constraint, Equation 3.3.1 imposes the constraint that exactly one inbound edge should exist in the robot path that brings the robot to the home vertex. The inbound constraint for the starting vertex s, Equation 3.3.2) of the path is set to 0 to ensure that there are no edges inbound to the starting vertex. In case the starting vertex is the home vertex, this constraint is not imposed.

$$\sum_{\substack{i=1\\i\neq h}}^{N} x_{i,h} = 1 \tag{3.3.1}$$

$$\sum_{\substack{i=1\\i\neq s}}^{N} x_{i,s} = 0 \qquad s \neq h \tag{3.3.2}$$

$$\sum_{\substack{i=1\\i\neq j}}^{N} x_{i,j} = y_j \qquad j \in \{1 \dots N\}, j \neq \{s, h\} \tag{3.3.3}$$

Equations 3.4.1 – 3.4.2 present the outbound constraints. Equation 3.4.1 shows the outbound constraint for the starting vertex. As the path must begin from the starting vertex, the outbound constraint is set to 1. The outbound constraint of the home vertex (see Equation 3.4.2) is set to 0 if the home vertex is different from the starting vertex. For all other vertices, the outbound vertex is set to 1 only if the vertex is a part of the path.

$$\sum_{\substack{i=1\\i\neq s}}^{N} x_{s,i} = 1 \tag{3.4.1}$$

$$\sum_{\substack{i=1\\i\neq h}}^{N} x_{h,i} = 0 \qquad h \neq s \tag{3.4.2}$$

$$\sum_{\substack{i=1\\i\neq j}}^{N} x_{j,i} = y_j \qquad j \in \{1 \dots N\}, j \neq \{s, h\} \tag{3.4.3}$$

Equation 3.5.1 presents the subtour elimination constraints based on the Miller Tucker Zemlin (MTZ) formulation [52]. It introduces a variable $u_i \in \mathbb{R}$ for each vertex i to store the order in which the vertices are visited.

$$\begin{aligned} u_i - u_j + 1 &\leq (N-1)(1-x_{ij}) && \forall i,j = 2,\ldots,N \\ 2 \leq u_i &\leq N && \forall i = 2,\ldots,N \\ u_i &\in \mathbb{R} \end{aligned} \tag{3.5.1}$$

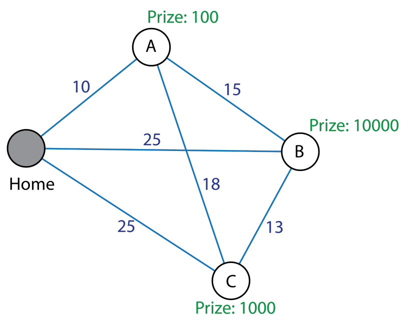
As the orienteering problem formulation seeks to maximize the collected prize within a given time budget or deadline constraint, it does not prioritize minimizing the cost of the path. In scenarios where the time budget exceeds the time required to visit all vertices, the orienteering problem tends to yield paths with higher costs. This issue is particularly prevalent during the initial stages of exploration when the robot's known map is small relative to the time budget. As a result, the orienteering problem formulation is suboptimal for smaller environments and larger time budgets.

For example, consider the complete graph in Figure 3.1, where the robot starts an orienteering tour from the Home vertex and ends its path at the Home vertex with a time budget of 50 time steps. The generated path visits the vertices: Home, B, A, Home with a path cost of 50 and a total prize collected of 10100. If the time budget is increased to 100 time steps, the generated path visits the vertices: Home, B, C, A, Home' with a path cost of 66 and a total prize collected of 11100. This path with a time budget of 100 time steps is not the shortest path to visit these vertices.

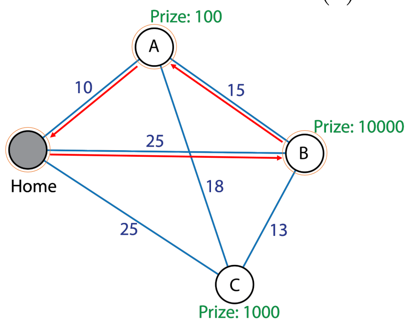
To address this issue and obtain the shortest prize path, we formulate a variant of the orienteering problem called the profitable tour problem.

3.4.2 Profitable Tour Formulation

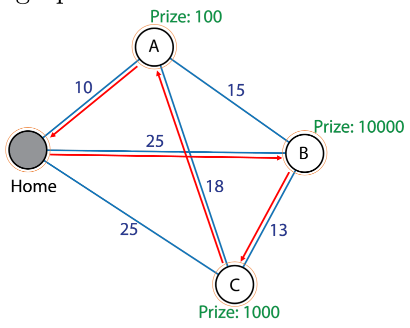
The Profitable Tour Problem (PTP) [45, 46] is a variant of OP. Its objective is to maximize the difference between the total collected prize and the cost incurred. For the single-robot indoor exploration problem, the PTP objective function is the scaled



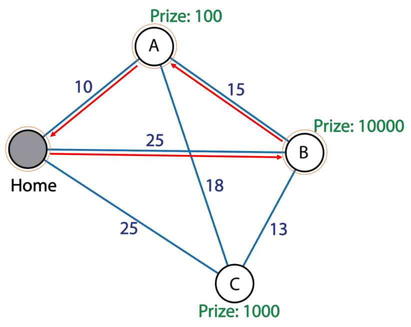
(a) Complete graph



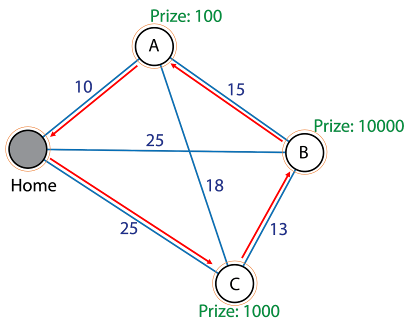
(b) Orienteering Problem ($t_r = 50$)



(c) Orienteering Problem ($t_r = 100$)



(d) Profitable Tour Problem ($t_r = 50$)



(c) Profitable Tour Problem ($t_r = 100$)

Figure 3.1: (a) illustrates an example of a complete graph. Consider that the robot is in the home position and it has three vertices to visit: A, B, and C with prizes 100,10000, and 1000 respectively. The edge costs are written in blue along each edge. (b) illustrates the path generated by the orienteering problem formulation with 50 time steps, (c) illustrates the path generated by the orienteering problem formulation with 100 time steps, (d) illustrates the path generated by the profitable tour problem formulation with 50 time steps, and (e) illustrates the path generated by the profitable tour problem formulation with 50 time steps. The path is shown as a sequence of red arrows.

sum of prizes over all the visited vertices minus the total cost (Equation 3.6), where m is a multiplier that scales the vertex prizes. The PTP formulation has the same constraints as the OP, including the deadline constraint.

$$\text{Maximize} \quad m \sum_{\substack{i=1 \\ i \neq s,h}}^{N} p_i \, y_i - \sum_{\substack{i=1 \\ i \neq h}}^{N} \sum_{\substack{j=1 \\ j \neq i,s}}^{N} c_{i,j} \, x_{i,j} \tag{3.6}$$

$$x_{i,j} \in \{0,1\}$$

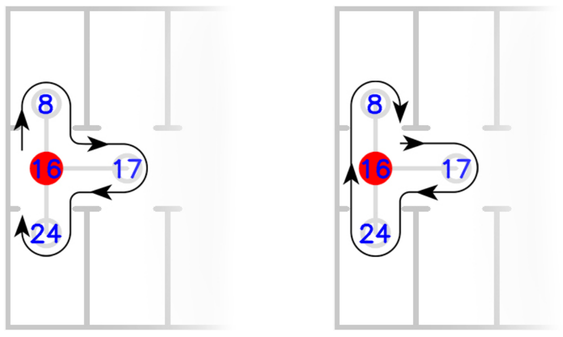
$$y_i \in \{0,1\}$$

where, m is a multiplier that scales the profit of all the vertices. We have used two values for m: 10 and 100.

The constraints to the above objective function is the same as the constraints for the orienteering problem. The time budget constraint is the same as in Equation 3.2, inbound constraints as shown in Equation 3.3.1 – 3.3.3, outbound constraints as shown in Equation 3.4.1 – 3.4.3, and subtour elimination constraints as shown in Equation 3.5.1.

The profitable tour problem formulation generates paths similar to those of the orienteering problem (OP) when the deadline is smaller than the time required to visit all vertices, as illustrated in Figure 3.1(d). Furthermore, the profitable tour problem formulation successfully computes the shortest path to visit the vertices, even when the deadline exceeds the time needed to visit all vertices. For example, in Figure 3.1(e), the solution to the profitable tour problem with a deadline of 100 steps results in a path with a cost of 63, whereas the orienteering problem formulation, shown in Figure 3.1(c), yields a path cost of 66.

Greedy Swap The path generated from the PTP and OP formulations is not formulated to make the first vertex in the path as a high-priority vertex such as a corridor. Instead it focuses on maximizing the total prize collected. However, in the problem of robot exploration, the robot visits the first vertex in the path created and updates the map. The existing path is generally not useful with the updated map,



(a) (b)

Figure 3.2: A straight corridor environment with the red circle showing the position of the robot. The numbers shows the index of the vertices. Consider the prize of vertices 8, 17, and 24 are 10, 10000, and 10 respectively. The cost of each edge is 2. (a) shows the possible solution path of [8, 17, 24, 16] created by the OP and PTP solver. As the robot will visit the first vertex of the path and recalculate the results, it would lead to a slower exploration. (b) shows the path [17, 8, 24, 16] after a greedy swap, where if a vertex has a higher prize, it will be moved up the list provided there is no increase in total path cost.

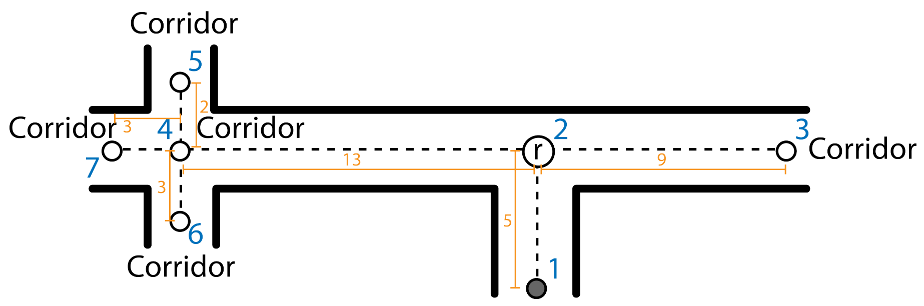
and a new path is recomputed accounting for the newly discovered vertices adjacent to the vertex visited. This means there exists situations, where the robot path contains a high priority vertex in the tour but it is not the first vertex, and it might be possible to have another path starting from the same vertex, where the same vertices are a part of the path, but the first vertex is a high priority vertex. One such situation is illustrated in Figure 3.2. In the figure, we observe that the path generated by the PTP solver is [8, 17, 24, 16]. The solution path visits all the discovered vertices in a shortest path. However, this path is not ideal as the robot visits vertex 8 as the first vertex, exploring the small room. Whereas, if the robot visited the corridor vertex 17, it could have spent the limited exploration time exploring the corridor which could have led to identifying a higher number of discovered vertices.

The procedure of greedy swap scans through the path generated by the solver, identifies vertices a set of vertices that have the same cost to visit from the current vertex of the robot. It swaps a vertex from this set with the first vertex of the path if two conditions are met. First, the swapped vertex has a higher priority, second, the resultant path is the same cost.

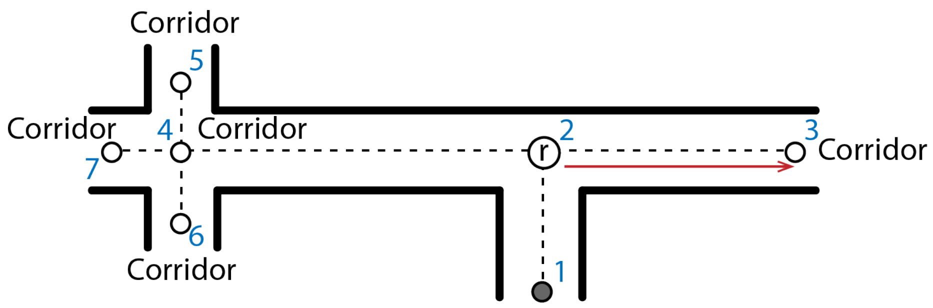
3.4.3 Minimum Latency Path

Minimum Latency Path(MLP) [53] Problem, also known as the Traveling Deliveryperson Problem (TDP) [47] or Cumulative Traveling Salesperson Problem [54] aims to minimize the average duration the robot takes to visit a vertex. Given a predefined start and goal vertex, the problem asks for an optimal sequence of vertices to be traversed in a path from the start to goal vertex. The objective is to minimize the cumulative path cost to all vertices. The cumulative path cost is analogous to “service time" at which a certain vertex is visited by the robot. Here “service time" refers to the walltime at which a vertex is visited by the robot. We consider that the robot doesn't require any additional time to “service" a vertex, however just visiting the vertices is enough to consider the vertex serviced.

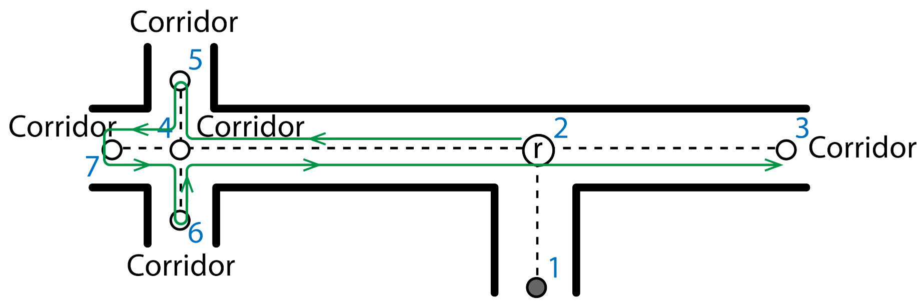
Modeling the prioritized robot exploration problem as a Minimum Latency Path(MLP) problem allows the identification of the shortest cumulative path cost to visit all frontier vertices. This approach allows the robot to prioritize traveling to a larger cluster of high-priority vertices rather than focusing on the closest high-priority vertex. An illustrative example is shown in Figure 3.3. In this example, observe that the Priority-based Greedy algorithm chooses to visit vertex 3 first, followed by the other vertices, as vertex 3 is closest to the robot. In contrast, the minimum latency path visits the left side of the corridor to maintain the lowest cumulative path cost to all vertices, consequently allowing the robot to keep the cumulative cost at $(15 + (15 + 5) + (15 + 5 + 6) + (15 + 5 + 6 + 25) = 112)$. By minimizing the cumulative cost, the average time to visit all frontier vertices is reduced. In this case, the average cost to visit each frontier vertex is $(112/4 = 28)$. Alternatively, if the goal was to minimize the total cost of the path, the path would have been to visit the vertices in order: (2, 3, 5, 6, 7). In this case, the cumulative cost would have been $9+(9+24)+(9+24+5)+(9+24+6) = 119$, making the average cost to visit each frontier vertex $(119/4 = 29.75)$. Our hypothesis for employing the minimum latency



(a) Weighted Graph



(b) Priority-based Greedy Path



(c) Minimum Latency Path

Figure 3.3: Comparison of Priority-based Greedy and Minimum Latency Path for an example scenario with several corridor vertex in a graph. (a) Shows the weighted graph with distance along each edges. Vertex 2 is the current position of the robot, the white vertices are discovered vertices, while the gray vertex 1 is a visited vertex. (b) Shows the path generated by the Priority-based greedy problem. Notice that the robot visits the closest vertex first and does not have a overhead to visit other vertices. (c) Shows the path generated by the Traveling Deliveryperson problem. Notice here that the robot tries to visit the cluster of corridor vertices on the left of the graph before visiting the vertex on the right.

path problem formulation is as follows: since visiting each frontier vertex provides an equal likelihood for the robot to explore a portion of the unknown environment, a dynamic deadline would favor a formulation with a lower average cost per frontier vertex, thereby increasing the probability of visiting a larger number of frontier

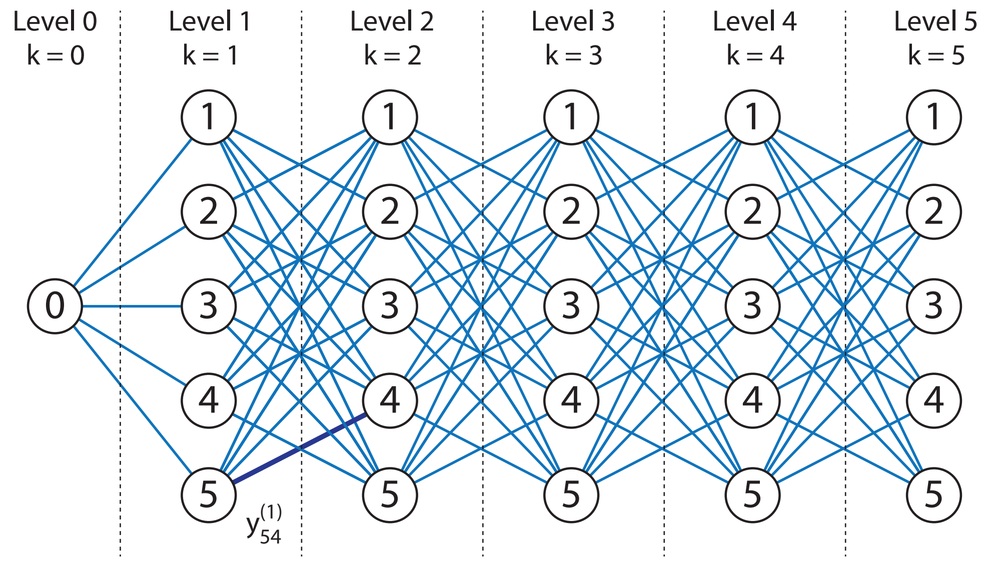


Figure 3.4: Graphical representation of the multi-level network. Each level corresponds to a value of k. Each edge is represented by a binary variable $y^{(k)}_{ij}$. Highlighted edge shows $y^{(1)}_{5\,4}$ as it starts from level 1 and is from vertex 5 to vertex 4.

vertices.

The MLP problem calculates the sequence of vertices the robot visits in a path. To calculate the sequence of vertices, considering vertex 0 as the first vertex of the path, a multi-level network can be constructed (see Figure 3.4), in which each level is a repetition of the same graph. Repeating the graph we can assign binary variables for each edge at every level. The level denotes the vertex sequence in the path. Each level allows for one vertex to be selected, and once a vertex is selected, it cannot be reselected in subsequent levels. This allows the robot to create a path to visit all of the vertices. The number of levels determine the lookahead of the path planning. The mathematical formulation is discussed below.

To calculate the sequence of vertices, considering vertex 0 as the first vertex of the path, we can formulate the mixed integer linear program with the objective function to minimize the cumulative path cost to each vertices visited along the path. The objective function can be written as Equation 3.7, where $y^{(k)}_{ij}$ is a binary variable which is 1 if vertices i and j are connected in the kth step of the sequence of vertices in the resultant path.

$$\min \qquad n\sum_{i=1}^{n} c_{0i}x_i^{(1)} + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}(n-k)c_{ij}y_{ij}^{(k)} \tag{3.7}$$

where,

- c_{ij} is the edge cost of the edge from vertex i to j
- k is the level
- $y_{ij}^{(k)} \in \{0,1\}$ is 1 if the edge from i to j is active in between levels k and $k+1$, 0 otherwise.
- $x_i^{(k)} \in \{0,1\}$ is 1 if a vertex $i \in V$ is visited at level k.

The constraints are:

- All levels needs to be occupied, but no more than two nodes can occupy a level

$$\sum_{k=1}^{n} x_i^{(k)} = 1 \qquad (i = 1, 2, \ldots, n) \tag{3.8}$$

- Guarantee that each level is occupied by a single node.

$$\sum_{i=1}^{n} x_i^{(k)} = 1 \qquad (k = 1, 2, \ldots, n) \tag{3.9}$$

- Ensure that one arc leaves from level k

$$\sum_{\substack{j=1\\ j\neq i}}^{n} y_{ij}^{(k)} = x_i^{(k)} \qquad (i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, n-1) \tag{3.10}$$

- Impose that at level $k+1$ only one arc can arrive at a time

$$\sum_{\substack{j=1 \\ j\neq i}}^{n} y_{ji}^{(k)} = x_i^{(k+1)} \qquad (i = 1, 2, \ldots, n;\ k = 1, 2, \ldots, n-1) \tag{3.11}$$

- Establish that $x_i^{(k)}$ are binary numbers

$$x_i^{(k)} \in \{0, 1\} \qquad (i = 1, \ldots, n;\ k = 1, 2, \ldots, n) \tag{3.12}$$

- Establish that $y_{ij}^{(k)}$ are binary numbers

$$y_{ij}^{(k)} \in \{0, 1\} \qquad (i = 1, \ldots, n;\ j = 1, \ldots, n;\ i \neq j;\ k = 1, 2, \ldots, n) \tag{3.13}$$

The formulation of minimum latency path problem does not consider the node prizes into account. To consider node prizes, we formulate a variant of the minimum latency path Problem and the profitable tour problem as discussed below.

3.4.4 Profitable Tour Problem with Minimum Latency Path

To account for the prizes of the vertices and to consider the minimum latency path, we formulate the prioritized exploration problem in another formulation, Profitable Tour Problem with Minimum Latency Path (PTP-MLP). Here, the objective of PTP-MLP formulation is to maximize the difference between the prizes collected by visiting vertices and cumulative cost while exploring vertices. The cumulative costs ensure that the average path cost to each vertex is minimized. In this formulation we also use discounted prizes to ensure that high priority vertices are visited earlier in the path.

The objective function for the PTP-MLP formulation is defined in Equation 3.14, where m is the multiplier and $p_i^{(k)}$ is the prize at vertex i at level k. The variables $x_i^{(k)}$ and $y_{ij}^{(k)}$ are binary. The constraints are slightly different from the constraints in

Minimum Latency Path Problem.

$$\max \qquad m\sum_{k=1}^{n}\sum_{i=1}^{n} p_i^{(k)}x_i^{(k)} - \left(n\sum_{i=1}^{n} c_{0i}x_i^{(1)} + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n}(n-k)c_{ij}y_{ij}^{(k)} \right) \tag{3.14}$$

$$x_i^{(k)} \in \{0,1\} \qquad (i=1,\ldots,n;\ k=1,2,\ldots,n)$$

$$y_{ij}^{(k)} \in \{0,1\} \qquad (i=1,\ldots,n;\ j=1,\ldots,n;\ i\neq j;\ k=1,2,\ldots,n)$$

The objective function is constrained by the total time constraint, Equation 3.15. The formulation considers that not all levels needs to be occupied, but no more than one nodes can occupy a level, as shown in Equation 3.16.

$$\sum_{i=1}^{n} c_{0i}x_i^{(1)} + \sum_{k=1}^{n-1}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} c_{ij}y_{ij}^{(k)} < B \tag{3.15}$$

$$\sum_{k=1}^{n} x_i^{(k)} \leq 1 \qquad (i=1,2,\ldots,n) \tag{3.16}$$

The Equation 3.17 ensures that at most one arc leaves from level k to $k+1$. This prevents allows for the path to remain in budget and stop at a vertex.

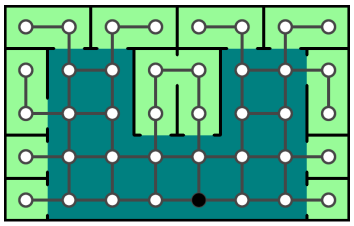
$$\sum_{\substack{j=1\\ j\neq i}}^{n} y_{ij}^{(k)} \leq x_i^{(k)} \qquad (i=1,2,\ldots,n;\ k=1,2,\ldots,n-1) \tag{3.17}$$

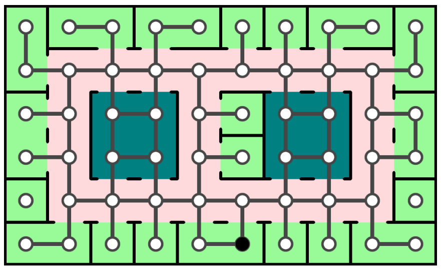
Similarly, Equation 3.18 ensures that there is at most one edge from each level k.

$$\sum_{i=1}^{n} x_i^{(k)} \leq 1 \qquad (k=1,2,\ldots,n) \tag{3.18}$$

Equation 3.19 imposes that at each level $k+1$, only one arc can arrive at a time.

$$\sum_{\substack{j=1\\ j\neq i}}^{n} y_{ji}^{(k)} = x_i^{(k+1)} \qquad (i=1,2,\ldots,n;\ k=1,2,\ldots,n-1) \tag{3.19}$$


(a) Large Home


(b) Research Lab

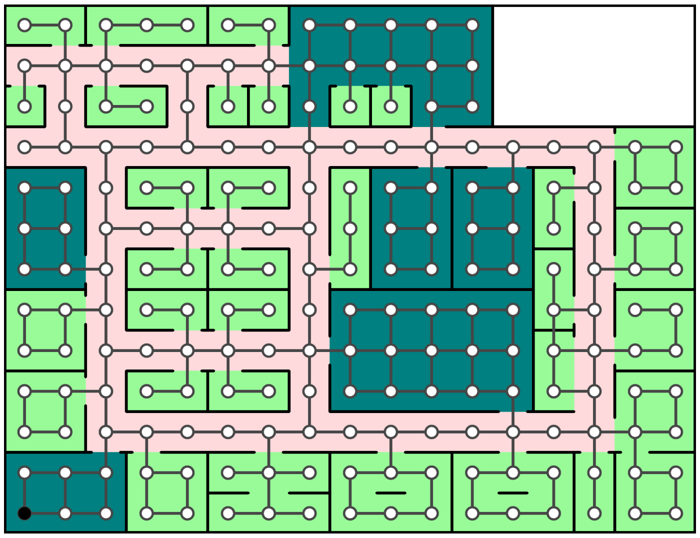

(c) Office

Figure 3.5: Graph environments for testing exploration algorithms with small rooms (light green), large rooms (dark green), and corridors (pink). The first row consists of two environments with different connectivity: (a) Large Home, which has no corridors, and (b) Research Lab, which has two looped corridors. The third row illustrates the (c) Office environment, which is the largest. The black vertex shows the starting vertex for exploration. All environments are drawn to scale.

3.5 Experiments

In this section, we present the simulation experiments conducted to evaluate the performance of the exploration algorithms. The experiments were carried out in two distinct types of environments: graph-based environments, and 3D environments in Gazebo.

The graph-based environments were employed to assess the efficiency of the exploration algorithms in isolation from the complexities of mapping, localization, and robot navigation. These environments provide a simplified representation of the exploration space, allowing for a more focused evaluation of the algorithmic performance. We use a total of six simulated environments. Three of these are elementary graph environments from Chapter 2. There are three more environments which are a combination of the elementary layouts shown in Figure 3.5. The Large Home (Figure 3.5(a)) has no corridors and consists of a large room with small rooms connected to it. The Research Lab (Figure 3.5(b)) has two looped corridors, and Office (Figure 3.5(c)), the largest environment, has multiple corridors with multiple intersections.

To complement the graph-based experiments, we conducted experiments in 3D environments using the Gazebo simulation platform. These experiments offer a more realistic setting for robot exploration, enabling the assessment of the algorithms in environments that closely mimic real-world scenarios. The repeatability of the Gazebo experiments ensures the reliability of the results obtained. A set of three environments, Straight Corridor, Looped Corridor, and Branched Corridor from [41] has been used to simulate real-world exploration (Figure 2.5). We replicate the experimental setup from Section 2.4.1.2), employing the three graph environments illustrated in Figure 2.5. Utilizing the Gazebo simulator [24], these environments are navigated by a simulated TurtleBot3 robot. The robot is equipped with a single scan 360° LiDAR, with a range of 6 meters, which simulates the Slamtec RPLiDAR A1M8 sensor. This setup allows for a consistent evaluation of the exploration algorithms across different environments.

The GMapping SLAM algorithm [25] is used to create an occupancy grid map from the sensor and odometry data. This occupancy grid map is converted to the skeleton graph as mentioned in Section 2.4.1.2. We convert the map to a skeleton graph. The exploration graph G is represented by this skeleton graph. We determine each vertex as a frontier vertex if it is possible to view the frontier cells from that particular vertex. We calculate this by ray tracing a 360° LiDAR from the vertex position. If the rays coincide with a frontier cell, the cell views a part of the frontier. If more than five percent of the rays from vertex coincide with frontier cells, the vertex is marked as a discovered vertex.

As the optimization based algorithms have worst-case exponential computational complexity, to make sure the algorithm runs in a sub-second runtime, once the number of discovered vertices in the exploration graph exceeds 15, the vertices are clustered using the k-medoids algorithm [55] into 10 clusters to ensure fast optimization solve times. The prize of a vertex cluster is the average prize of each vertex in the cluster.

3.5.1 Metrics

The performance of the exploration algorithms is evaluated using the percentage of the environment explored. For the graph environments, we compare the performance of the (1) Priority-based greedy exploration (P-Greedy) algorithm from [41], (2) Cost-based greedy (C-Greedy) algorithm, the baseline exploration algorithm motivated by [1], (3) OP-based exploration algorithm (OPE), (4) PTP-based exploration algorithm (PTPE), (5) MLP-based exploration algorithm (MLPE), and (6) PTP-MLP based exploration algorithm (PTP-MLPE). The P-Greedy algorithm visits the highest priority vertex that is feasible given the deadline. It prioritizes in the order of corridors, large rooms, and small rooms. The C-Greedy algorithm visits the closest vertex that is feasible given the deadline, regardless of the priority of the vertex. Furthermore, we report the computation time for the exploration algorithms. For the Gazebo environments, we compare the P-Greedy, OPE, and PTPE algorithms. We test the Gazebo environments with a limited set of exploration algorithms due to the large computation time of MLPE and PTP-MLPE algorithms.

For the graph environments, explored regions of the map are represented by the set of explored vertices V, where $V = (V_d \cup V_{vis})$. The performance metric for the graph environments is the percentage of explored vertices in the complete environment. The performance metric for the Gazebo environments is the percentage of the total area explored.

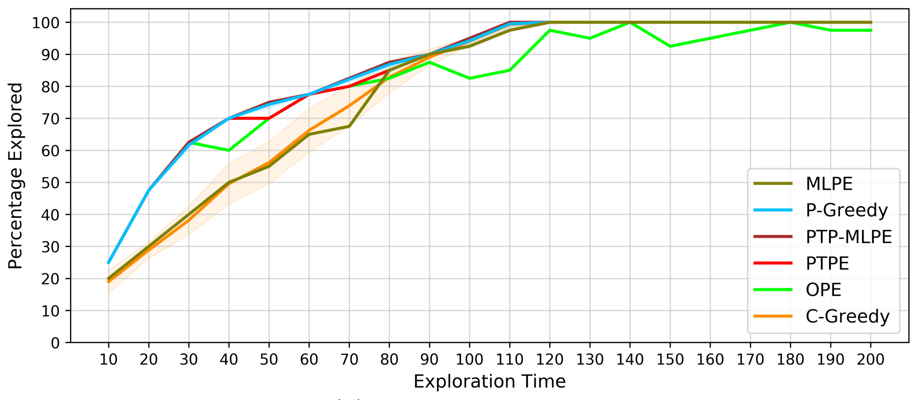
In the Gazebo environments, exploration time is measured in seconds and includes computation time and the time taken for robots to move in the environment. In all experiments, the robots are informed of a deadline at the start of exploration.

3.6 Results

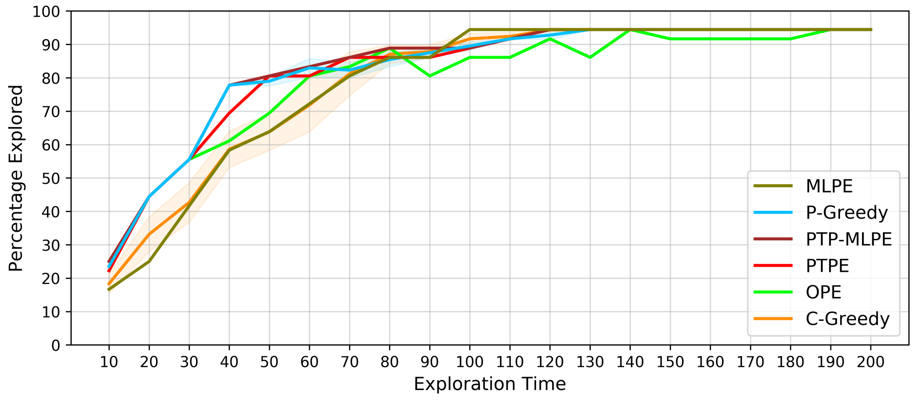
We compare the performance of the algorithms OPE, PTPE, MLPE, and PTP-MLPE to the priority-based greedy exploration algorithm (P-Greedy) [41]. For the OPE and PTPE, the prizes of the vertices in corridors, large rooms, and small rooms are set to 10000, 100, and 1, respectively.

The algorithms (P-Greedy, C-Greedy, OPE, and PTPE with multiplier $m = 10$) have been tested on the six graph environments shown in Figure 2.4 and 3.5. The prioritized exploration problem requires two parts of exploration as shown in to Figure 2.1: exploration without deadline and exploration with deadline. The orienteering problem formulation and the profitable tour formulation requires a deadline for the formulation. Both the OP formulation and PTP formulation uses the priority-based greedy algorithm as shown in Chapter 2 for exploration without deadline. We compare the percentage of the environment explored by the different exploration algorithms within a deadline. The experiments were run on an Intel Core i7 9800X with 64 GB of RAM, using the Python (v3.6.8) wrapper of Gurobi Optimizer (v9.5.1).

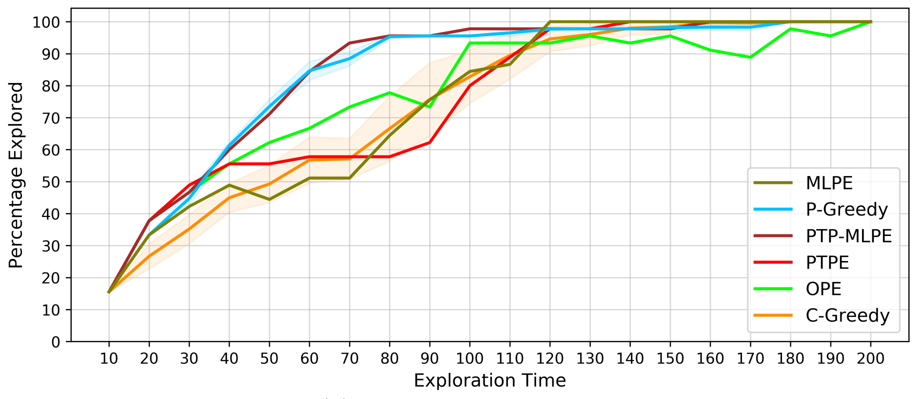
Additionally, we evaluate the computation time for various exploration algorithms across the six distinct graph environments shown in Figure 2.4 and 3.5. Computation time is defined as the total duration required to complete all computational steps involved in the exploration process. These results are illustrated in Figures 3.9 and 3.10. Notably, the greedy algorithm exhibits significantly lower computation times compared to those of the optimization-based exploration algorithms. Among the algorithms assessed, the Priority-based Greedy (P-Greedy) algorithm demonstrates the most efficient performance, yielding the shortest computation times relative to the percentage of the graph explored. Conversely, the PTP-MLPE and the MLPE algo-



(a) Straight Corridor



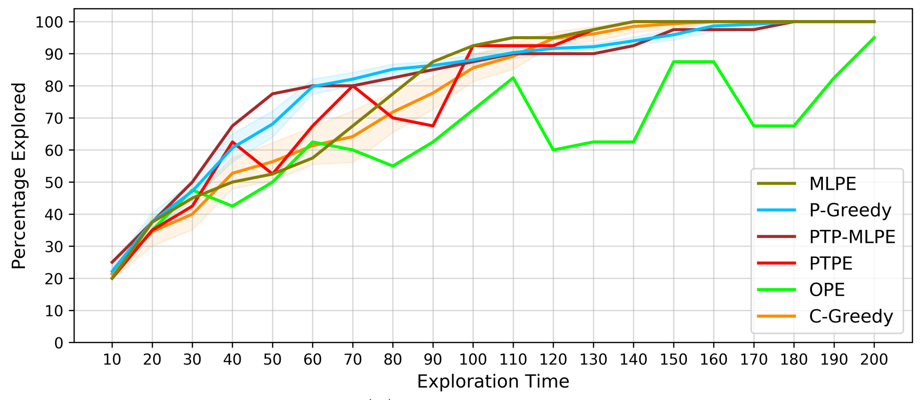
(b) Looped Corridor



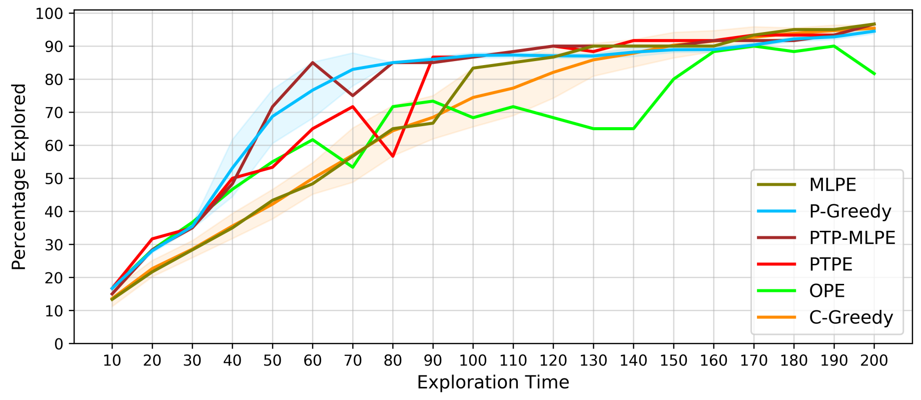
(c) Branched Corridor

Figure 3.6: Plots comparing the performance of the P-Greedy, C-Greedy, OPE, and PTPE, Minimum Latency Path based exploration (MLPE) and Profitable Tour Problem with Minimum Latency Paths based exploration (PTP-MLPE) prioritized exploration algorithms for the graph environments of Figure 2.4. The vertical axis shows the average percentage of explored vertices over thirty independent exploration trials for each deadline. The horizontal axis shows the deadlines. The shaded region shows the standard deviation of thirty independent experiments.

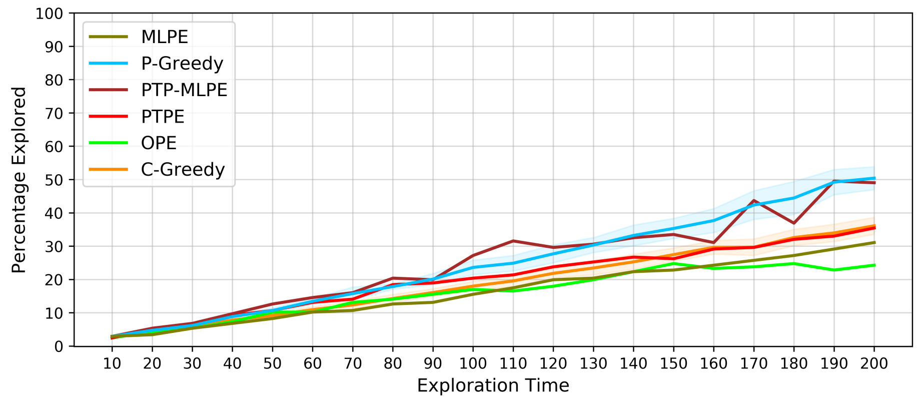
rithm has the highest computation time in most of the exploration algorithms. Due to their large computation time, these algorithms are deemed unsuitable for deployment



(a) Large Home

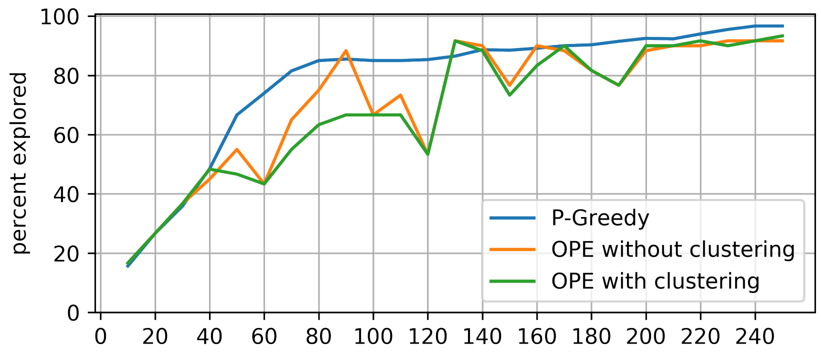


(b) Research Lab

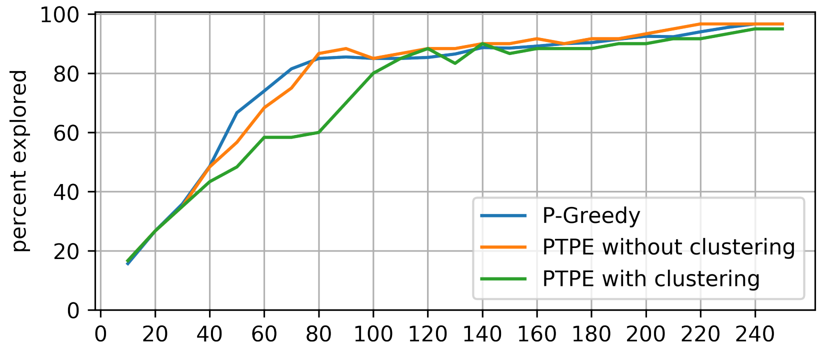


(c) Office

Figure 3.7: Plots comparing the performance of the P-Greedy, C-Greedy, OPE, and PTPE, Minimum Latency Path based exploration (MLPE) and Profitable Tour Problem with Minimum Latency Paths based exploration (PTP-MLPE) prioritized exploration algorithms for the graph environments of Figure 2.4. The vertical axis shows the average percentage of explored vertices over thirty independent exploration trials for each deadline. The horizontal axis shows the deadlines. Note that the Office environment requires a larger time to explore due to its larger size. Within this deadline, the robot starts from the home location, explores the environment, and returns to the home location.



(a)



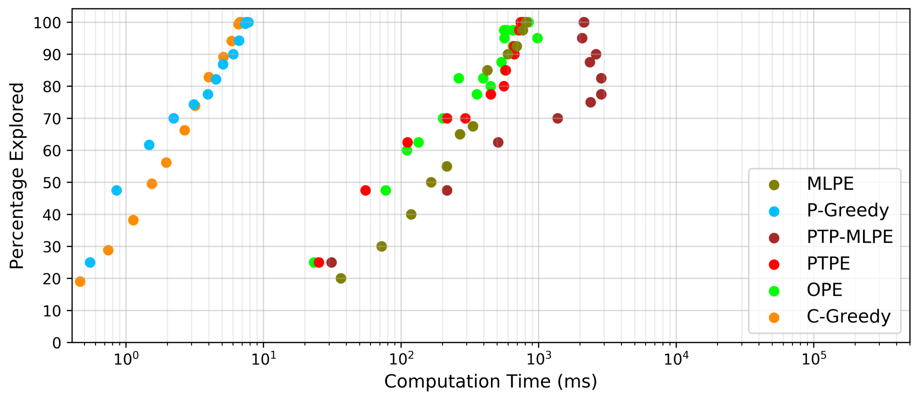
(b)

Figure 3.8: Impact of clustering on prioritized exploration of the Research Lab graph environment. The horizontal axis shows the deadlines. The plots show performance of the (a) OPE and (b) PTPE exploration algorithms without clustering (orange) and with clustering (green).

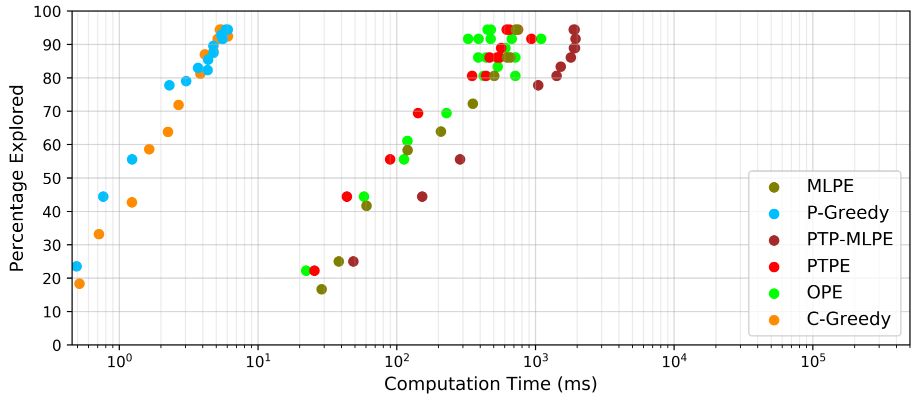
in Gazebo exploration scenarios.

To keep Gurobi's solve time under one second for the large graph environments, we clustered the discovered vertices into 10 clusters when the number of vertices exceeded 15. Clustering the vertices in the exploration environment reduces performance, as shown in Figure 3.8. For PTPE and OPE, the difference in the percentage of explored vertices with and without clustering is significant when the number of discovered vertices is greater than 2–3 times the number of clusters.

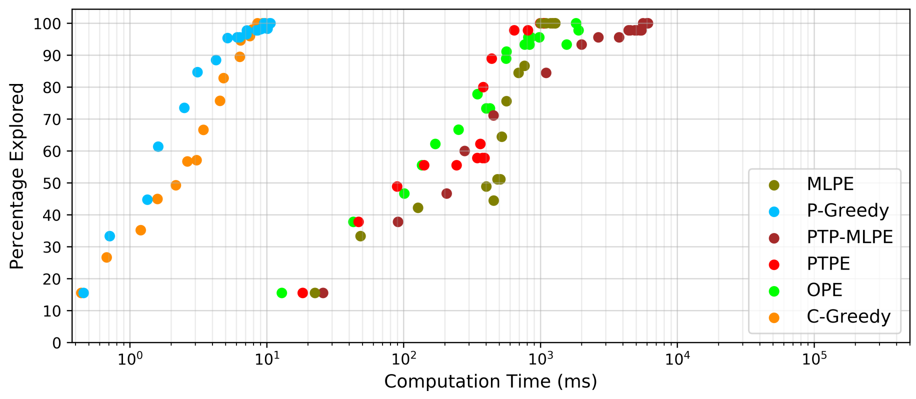
The performances of the P-Greedy, OPE, and PTPE algorithms on the three Gazebo environments of Figure 2.5 are compared in Tables 3.1, 3.2, and 3.3. For most deadline instances, the performances of all three algorithms are very close. For a few instances with performance differences, the causes can be identified. For example, the significant difference between P-Greedy and the optimization based algorithms



(a) Large Home



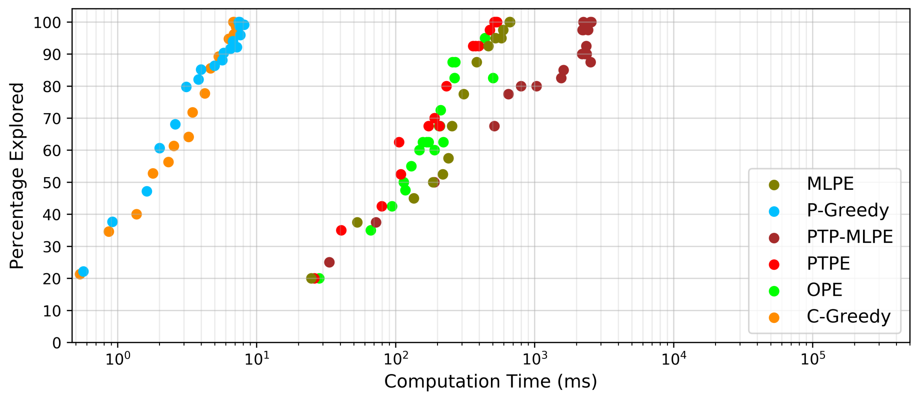
(b) Research Lab



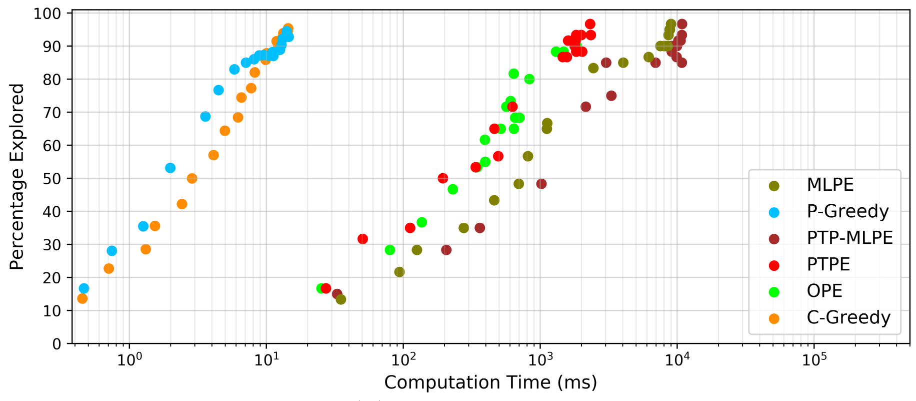
(c) Office

Figure 3.9: Semi-log plots comparing the compute time of the P-Greedy, C-Greedy, OPE, and PTPE, Minimum Latency Path based exploration (MLPE) and Profitable Tour Problem with Minimum Latency Paths based exploration (PTP-MLPE) prioritized exploration algorithms for the graph environments of Figure 2.4. The vertical axis shows the average percentage of explored vertices over thirty independent exploration trials for each deadline. The horizontal axis shows the total computation time to explore the environment.

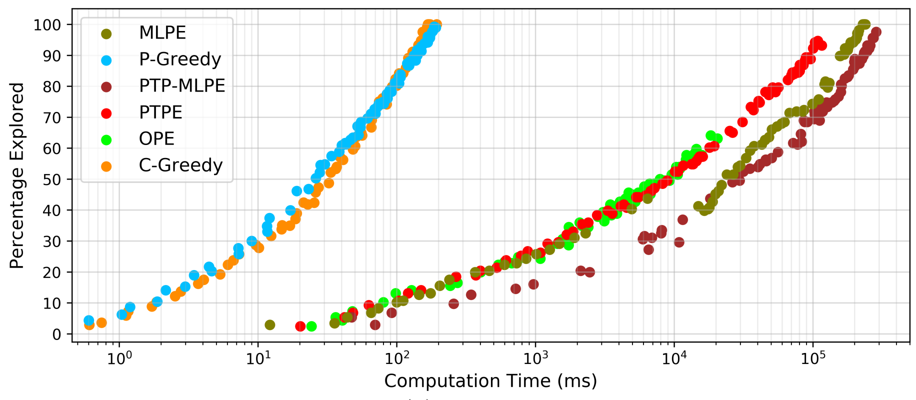
for the 1500 s deadline on the Straight Corridor environment stems from the skeleton graph having multiple vertices along the direction from the robot's position to the



(a) Large Home



(b) Research Lab



(c) Office

Figure 3.10: Semi-log plots comparing the compute time of the P-Greedy, C-Greedy, OPE, and PTPE, Minimum Latency Path based exploration (MLPE) and Profitable Tour Problem with Minimum Latency Paths based exploration (PTP-MLPE) prioritized exploration algorithms for the graph environments of Figure 2.4. The vertical axis shows the average percentage of explored vertices over thirty independent exploration trials for each deadline. The horizontal axis shows the total computation time to explore the environment.

frontier. See Section 3.7.2 for details.

Table 3.1: Exploration results for the Straight Corridor Gazebo environment as shown in Figure 2.5(a). P-Greedy is the prioritized-greedy algorithm, OPE is the OP-based exploration algorithm, and the PTPE is the PTP-based exploration algorithm with multiplier m = 10.

Deadline (in secs.)	Algorithm	Corridor	Large Room	Small Room	Total
1500	P-Greedy	99.3%	43.7%	97.0%	72.3%
	OPE	99.9%	63.9%	98.4%	82.5%
	PTPE	99.8%	80.0%	97.9%	89.8%
1000	P-Greedy	99.6%	16.3%	96.4%	59.2%
	OPE	99.4%	24.2%	97.4%	63.4%
	PTPE	99.8%	28.5%	97.1%	65.3%
500	P-Greedy	79.6%	0.0%	78.5%	41.8%
	OPE	80.3%	0.0%	52.6%	31.9%
	PTPE	79.3%	0.0%	51.7%	31.4%

3.7 Discussion

The OPE and PTPE algorithms create paths to visit the set of vertices that maximize their objectives while ensuring the robot satisfies the deadline. In both formulations, there is no guarantee that the first vertex in the path will be a high priority vertex. As the path is recomputed after the robot visits a discovered vertex, the new path may again have the same limitation. The P-Greedy algorithm, in contrast, chooses the highest priority vertex that is closest to the robot's current position as the target vertex, provided the robot can explore the vertex and return home within the deadline. Given this fundamental difference between the P-Greedy algorithm — a short horizon exploration algorithm and the non-greedy PTPE and OPE algorithms — long horizon exploration algorithms, this section discusses our observations from executing these exploration algorithms on different types of environments.

3.7.1 Exploration in the Graph Environments

The P-Greedy algorithm has the best performance for most deadline instances, as shown in Figures 3.6 and 3.7. PTPE is second in performance in the Straight Corridor,

Table 3.2: Exploration results for the Looped Corridor Gazebo environment as shown in Figure 2.5(b). P-Greedy is the prioritized-greedy algorithm, OPE is the OP-based exploration algorithm, and the PTPE is the PTP-based exploration algorithm with multiplier m = 10.

Deadline (in secs.)	Algorithm	Corridor	Large Room	Small Room	Total
1500	P-Greedy	99.9%	99.7%	98.4%	99.1%
	OPE	100%	99.7%	99.1%	99.6%
	PTPE	100%	99.7%	94.6%	97.2%
1000	P-Greedy	100%	99.9%	97.2%	98.0%
	OPE	100%	99.7%	90.0%	94.9%
	PTPE	100%	99.5%	96.2%	98.0%
500	P-Greedy	80.3%	97.9%	51.9%	68.3%
	OPE	87.4%	98.8%	50.6%	69.5%
	PTPE	79.9%	97.7%	49.3%	66.0%

Looped Corridor and Research Lab environments. For a few deadline instances, PTPE outperforms P-Greedy on the Looped Corridor and Research Lab environments. In the Branched Corridor environment, P-Greedy significantly outperforms the other algorithms. Both P-Greedy and PTPE initially explore one of the two available corridors. Once the corridor is visited, P-Greedy chooses to visit the other corridor, while PTPE explores the room vertices adjacent to the explored corridor instead of visiting the other corridor, hurting its performance. In the Large Home environment, OPE performs better than PTPE for smaller deadline instances.

3.7.2 Exploration in the Gazebo Environments

The Gazebo environments use skeleton graphs. A skeleton graph updated after the robot reaches a target vertex may not have a vertex at the current robot position. As a result, the robot spends a few seconds orienting and moving to the new skeleton graph, impacting the exploration time. Furthermore, a shortest path along the skeleton graph might not be the shortest path between two points on the map. As shown in Figure 3.11, the skeleton graph may generate multiple vertices between the robot's

Table 3.3: Exploration results for the Branched Corridor Gazebo environment as shown in Figure 2.5(b). P-Greedy is the prioritized-greedy algorithm, OPE is the OP-based exploration algorithm, and the PTPE is the PTP-based exploration algorithm with multiplier m = 10.

Deadline (in secs.)	Algorithm	Corridor	Large Room	Small Room	Total
1500	P-Greedy	76.3%	63.2%	57.3%	64.0%
	OPE	76.6%	77.7%	50.3%	65.3%
	PTPE	75.0%	60.7%	57.1%	62.8%
1000	P-Greedy	66.8%	54.7%	49.2%	55.5%
	OPE	66.1%	57.0%	36.9%	50.5%
	PTPE	71.5%	49.9%	47.6%	59.8%
500	P-Greedy	50.2%	24.2%	23.9%	30.9%
	OPE	55.1%	32.8%	22.4%	34.1%
	PTPE	55.1%	28.6%	27.9%	35.3%

current position and the frontier of the explored map. The P-Greedy algorithm visits the highest priority vertex that is closest to the robot, while OPE and PTPE may visit a different target vertex, farther away from the robot's current position. This results in frequent skeleton graph updates for the P-Greedy algorithm, and each time, the robot spends a few seconds repositioning itself in the new skeleton graph, thereby affecting the P-Greedy exploration performance.

3.7.3 Limitations of Online Methods

The exploration problem can be categorized as an online problem [56] as the robot sequentially receives information from the initially unknown environment. As the exploration environment is partially observed by the robot, optimal solutions for the partially known environments might not be optimal over the entire environment. We notice that each exploration algorithm performs differently for a given environment. As the structure of corridors and rooms determine the performance of the algorithm, we can consider the environment as an oblivious adversary [57]. However, if an adversary is aware of the exploration algorithm's priorities for visiting different building

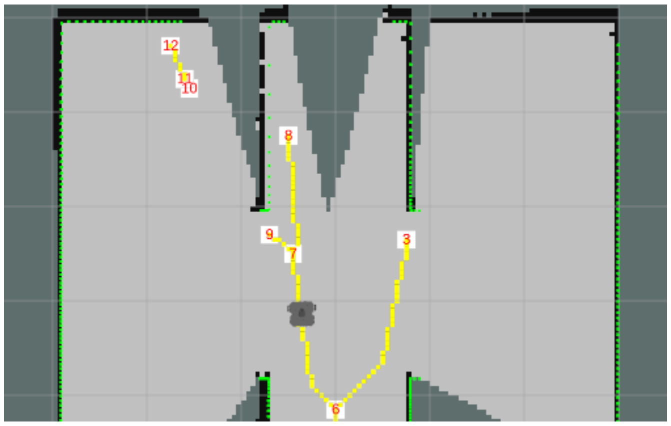


Figure 3.11: Robot exploring the Straight Corridor Gazebo environment, moving along an edge of the skeleton graph. Skeleton graphs in Gazebo environments generate multiple vertices along the direction from the robot's current position to the exploration frontier. Here the vertices (7, 9, 8) are along the corridor. The skeleton graph shown here will be updated using the updated map after the robot reaches the target vertex. The frontier is the boundary between the light gray and dark gray regions of the map.

structures, it can devise an environment in which the algorithm would perform poorly. Although no particular exploration algorithm is a clear winner in all types of environments, we observe that the priority-based greedy algorithm performs competitively in almost all environments.

3.8 Contributions

In this chapter, we formulated an exploration problem as an Orienteering problem, Profitable Tour problem, Minimum Latency Path problem, and Profitable Tour problem with minimum latency path. These formulations enable a lookahead in planning, allowing for decision-making beyond just the next step.

3.9 Conclusion

This chapter considered the prioritized exploration problem, whose objective is to compute the geometric layout of an initially unknown environment by rapidly exploring it and returning to the home location within a deadline. We formulated it as an Orienteering Problem and as a Profitable Tour Problem. In the graph environments, we found that a priority-based greedy exploration algorithm performs on par or better than the optimization based algorithms in most instances. In the Gazebo environments, all three exploration algorithms perform very similarly. While no algorithm emerged a clear winner across all exploration environments, the priority-based greedy algorithm performed quite competitively despite its low computational overhead.

We plan to investigate two directions in our future work. The first is further testing of the prioritized exploration algorithms in Gazebo and real-world environments to better understand their performance. This includes tests over a large set of deadlines and environments, and using a separate path planner to compute efficient collision-free trajectories for faster robot exploration. The second direction is to obtain the values of the vertex priorities based on the semantic information they provide about the building structure.

CHAPTER 4: MULTI-ROBOT PRIORITIZED INDOOR EXPLORATION WITH DYNAMIC DEADLINES

In the past decade, mobile robots have observed a steep decrease in prices, as robots have been made more commonplace. However, mobile robots have limitations of limited battery life, sensing, computation, and communication capabilities. These limitations restrict the capabilities of an individual robot to solve a large problem. This leads to a growing need for algorithms and planning to make multiple mobile robots cooperatively coordinate together to achieve a single large task [58].

Exploration with deadlines is useful in time-critical and dangerous environments when the computed environment layout can provide essential information for subsequent robotic or manned operations. It can also be used to identify hazardous regions of an environment. When the robots have insufficient time to explore the entire environment, a single-robot prioritized exploration algorithm as shown in Chapters 2 and 3 can explore highly connected areas of the environment to obtain the layout of the environment.

In this chapter, we improve on single-robot exploration by using multi-robot coordination to explore an initially unknown indoor environment. In order to increase the amount of explored area within the dynamic deadline imposed by the environment or the robots, we can use multiple robots to explore the environment. As each robot is equipped with a set of sensors to sense and map the environment, using multiple robots mean that multiple robots can observe the unexplored area concurrently, leading to larger fraction of the environment being explored within a limited time. Multiple robots offer higher reliability as a subset of the functional robots can complete the exploration even if a few robots are deemed unusable. Robots are deemed unusable when they run out of battery, are trapped due to moving obstacles, or are just malfunctioning. When using multiple robots, it is also possible to use robots of different types. This makes it more suitable for exploration of complex environments, where one type of robot may be unsuitable to explore the complete environment within a limited time.

Most single and multi-robot exploration algorithms focus on exploring the complete environment instead of rapidly exploring the unknown environment to identify its layout. Since using multiple robots would allow for more extensive exploration when given a short deadline, this chapter is motivated by deadline driven multi-robot exploration, a topic that has received a limited amount of research attention.

Our research contributions are: (1) We introduce a multi-robot deadline-driven priority-based exploration algorithm; (2) We present a multi-robot algorithm for the Profitable Tour Problem based on a Team Orienteering Problem approximation algorithm; (3) We introduce preemptive exploration in our prioritized exploration algorithms, making the robot exploration more efficient.

The chapter discusses the background on multi-robot systems in Section 4.1, relevant literature in Section 4.2 and describes the multi-robot priority-based greedy algorithm and adaptation of the multi-robot Profitable Tour Problem in Section 4.3. Results of simulation experiments along with a discussion of the behavior of the exploration algorithms are presented in Section 4.4.

4.1 Background

Multi-robot systems (MRS) has several applications using mobile robots. MRS has been used for coverage problems to coordinate a fleet of drones by Bartolleni et. al. [59] for outdoor inspection of several points and by Saurav et. al. [60] for outdoor inspection of lines. MRS has been used for search and rescue problems [61, 62], formation control [63, 64].

A Multi-robot System (MRS) is defined as a robotic solution or technique that in-

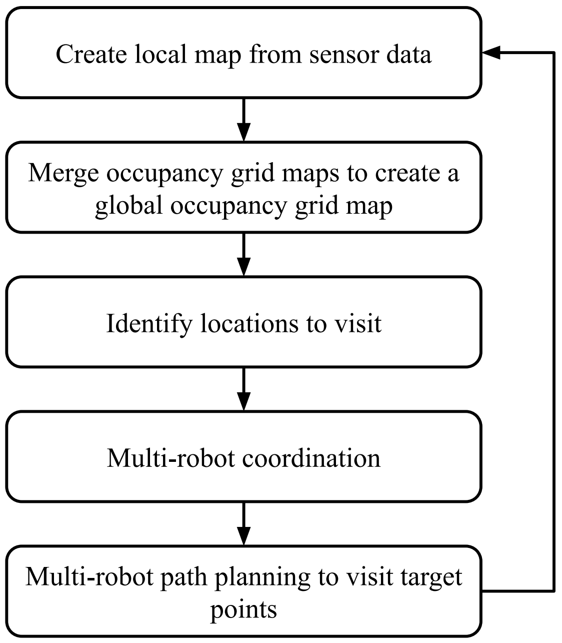

Create local map from sensor data
Merge occupancy grid maps to create a global occupancy grid map
Identify locations to visit
Multi-robot coordination
Multi-robot path planning to visit target points

Figure 4.1: Steps of most multi-agent exploration algorithms using a centralized controller. Notice that the algorithms do not direct the robots to home location after the exploration is complete.

volves multiple robots to solve a task in a cooperative way. MRS was first used in the problem of robot exploration by Rekletis [65], to specifically address the problem of exploring large environments. Rekletis used individual robot sensor data and tracked robot position from other robots to sweep the unknown area in parallel line trajectories. This was later significantly improved by Stachniss [66] to include techniques such as semantic place labels to improve exploration, and actively close loops during exploration to make the explored map more robust to mapping errors.

4.1.1 Multi-robot exploration

The steps of Multi-robot exploration can be divided into five steps as shown in Figure 4.1. Here we discuss about each of the steps and the related work associated with them.

Create local map from sensor data: Individual robots use simultaneous localization and mapping (SLAM) algorithms such as extended Kalman filter [7] to create a map of the environment surrounding the robot. This map is represented as an occupancy grid map [21], coverage maps [67] or sparse maps such as quadtrees and octrees [68]. These maps are discrete and are updated using a probabilistic update of the mapping algorithm.

Merge occupancy grid map to create a global map: By merging multiple local maps into a global map, the robots can be localized with respect to each other, and the robot exploration can be coordinated accurately. However, merging local maps into a single global map has additional challenges. As robots explore through an environment, odometry errors accumulate in the local map of the robots. These odometry errors mean the individual local maps do not exactly align one over the other. To address such a challenge, the robots must visit a common part of the environment, mark it as a landmark, as use it to align the local map correctly to one another. Existing techniques to merge individual maps from each robot utilize laser scans [69], pose matches [70], landmarks [71], and indoor features such as doors and corridor junctions [72].

Identify locations to visit: The set of robots identify a possible set of locations to visit to explore the map further. Ideally the set of locations would be at the periphery of explored and unexplored parts of the map, also known as frontier regions.

Multi-robot coordination: The goal of multi-robot coordination is to optimally subdivide the task to the participating robots, such that a robot does not remain idle or redo a previously completed task. Multi-robot coordination can be considered as the problem of multi-agent task allocation. Such task assignments may be done either through a centralized server or distributed through individual robots. The different types of coordination is illustrated in Figure 4.2. Existing techniques include greedy algorithms [5], decision theoretic algorithms [73, 74, 75, 2], segmentation-based algorithms [76, 77, 10, 78], and auction algorithms [79, 80, 81, 82].

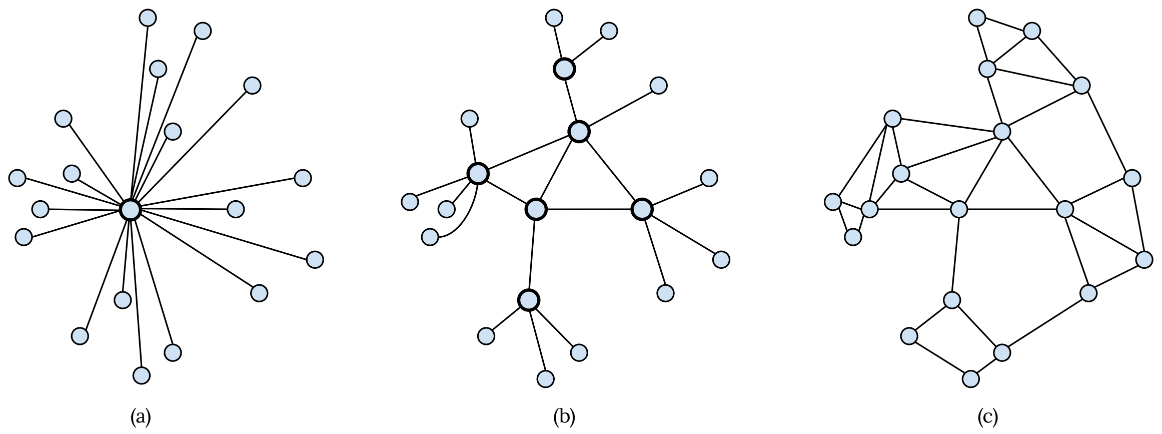


Figure 4.2: Coordination among agents in a general multi agent scenario. **(a)** Centralized control, where each individual agent is connected to the central controller (large circle in center). **(b)** Decentralized control, which is characterized by multiple compute units (circles with thicker border), that communicate with one another. Each compute unit acts as a local leader and coordinates the robots in its group. **(c)** Distributed control, which is characterized by local interaction between neighbors.

Multi-robot path planning Path planning of multiple robots [83] include collision avoidance with obstacles as well as other robots. The problem of path planning in static environment is a well studied problem. There are techniques such as planning in c-space, graph search techniques such as Dijkstra's algorithm [84], roadmap methods such as visibility graph, voronoi graph, and cell decomposition. Non-deterministic methods such as sampling methods such as Rapidly-exploring Random Trees(RRT) [85] and Probability Roadmap [86]. However, all of the methods work with static obstacles. To work with dynamic environments to avoid other moving robots, the robots should follow real-time motion planning. A few such techniques are Extended RRT (ERRT) [87], Artificial potential functions (APF) [88], graph search algorithm such as D* Algorithm [89, 90], Real-time Adaptive Motion Planning [91, 92], and even machine learning techniques [93].

4.1.2 Challenges

While MRE is more useful for handling larger environments than single robot systems, it brings its own set of challenges. These challenges are as follows:

Multi-robot coordination: The goal of multi-robot exploration is to achieve a functional autonomy with the least amount of time spent in overhead control, and optimally subdivide the task to the participating robots, such that a robot does not redo a pre-completed task. Algorithmically, multi-robot exploration can be considered as a task of optimally allocating specific regions to be explored by each robot. This can be expressed as the problem of dividing or partitioning the set of robots into non-overlapping sub-teams to perform the given tasks. It is mathematically equivalent to the well-known NP-hard problem of set-partitioning in combinatorial optimization [94]. Such task assignments may be done either through a centralized server or distributed through individual robots. Additionally, this step must also be done within a reasonable amount of time such that the robots do not stall during exploration.

Network and communication limitations: Wireless networks used to communicate with robots have limited bandwidth. Due to this, robots may not share their entire sensor information over the network. Each robot is required to pre-process their sensor information into a map.

For our problem, we assume that the robots can communicate among one another to send position and map information. Based on the prioritized exploration approach discussed in Chapter 2, we also know how to identify the location of frontier locations that lead to further exploration in the environment. We also should ensure that our proposed algorithm executes within a reasonable amount of time so that the exploration is not stalled due to long computation time.

4.2 Related Work

The problem of robot exploration has been studied from multiple perspectives. The work described in this chapter builds on previous work on robot exploration, trajectory planning and mapping techniques.

Early work in multi-robot exploration was performed by Rekleitis [65] and Yamauchi [5].The goal of efficient multi-robot exploration has been to direct the robot to non-overlapping areas of the exploration environment. It has been implemented through decision theoretic approaches developed based on visibility graphs [73, 74], environment segmentation [76], and next-best view based path planning approaches [95, 96]. The first multi-robot exploration algorithm which utilized semantic information of the environment was introduced by Stachniss [2]. A significant part of multi-robot exploration has focused on handling distributed robot exploration where it is expected that robots may not share a network to communicate with one another [97, 75, 81, 17, 64]. However, only a few techniques have considered exploration with battery constraints [15, 17]. Recent heterogeneous multi-robot exploration has been shown to collaborate for long periods of time switching between central coordination and distributed coordination depending on network availability [98]. Prioritized robot exploration with mission constraints have been discussed by [99], where the goal has been to utilize adaptive sensor coverage for time bound exploration.

The task of multi-robot exploration can be viewed as a variant of the Multiple Traveling Salesperson Problem (MTSP) [100] as each robot must compute a Hamiltonian path to visit the frontier locations to explore the map. Similarly, the Team Orienteering Problem (TOP) and Orienteering Problem with Neighborhoods, variants of the prize-collecting TSP with budget constraints, have been used to address the multi-robot exploration problem [39, 40]. However the Generalized TOP technique [39] has fixed rewards for exploring every unknown location, making it unsuitable for prioritized exploration.

The multi-robot exploration problem has been studied in the context of collaboration among robots and task scheduling [101]. While preemptive task scheduling has been used in multi-robot task allocation for warehouse logistics [102], it has not been investigated in the context of robot exploration.

The previous techniques have focused on exploration with the goals of exploring the complete environment or exploring under communication constraints. The single-robot prioritized exploration problem that we have addressed in Chapters 2 and 3 considers partial exploration of the environment within a changing deadline to explore the environment.

4.2.1 Limitations of existing work

The focus of the existing literature on multi-robot exploration has geared towards the faster exploration of the environment.

- Most methods do not consider deadlines from the robot or the environment.
- The current methods do not consider identifying the layout of environment as the objective of prioritized exploration.
- Most multi-robot exploration techniques do not consider return to home as a part of the exploration process.

4.3 Problem Formulation

A team of robots, represented by the set K, explores an unknown environment, senses and maps the area, and models the map as an exploration graph, $G = (V, E)$. V is the set of explored vertices, where each vertex $v \in V$ represents a physical location in the exploration environment. Each vertex is labeled based on the building structure it is located in, such as 'Corridor', 'Large Room', and 'Small Room'. Each vertex is assigned a non-negative prize p_i that represents its priority. The priority is defined by the building structure the vertex is located in. The set of vertices V is divided into two subsets V_d, a set of discovered vertices and V_{vis}, a set of visited vertices. Discovered vertices are based in physical locations explored by the robot but that have not been visited yet. Discovered vertices are located at the frontier of explored and unexplored areas. If a robot visits a vertex $v_i \in V_d$, it expects to observe a part of the unknown environment. E is the set of edges, where $e_{ij} \in E$ is an edge that connects two vertices v_i and v_j. An edge represents a collision free path between the two vertices. Each edge has a non-negative weight w_{ij} which is the estimated time to traverse the path represented by the edge. The robots start exploring the initially unknown environment with or without an imposed deadline t_r. As the robots explore, a deadline t_r is imposed on the robots. The robots should return to the home vertex v_h, from where it starts exploration, by the deadline t_r. The deadline t_r may change during exploration based on robot battery life or environmental parameters.

The exploration algorithm assigns a target vertex v_t^k to the kth robot, where $k = 1, 2, \ldots, |K|$. The exploration map is updated based on the sensor data from each robot k en route to v_t^k. The exploration graph is updated as soon as the robot reaches v_t^k. Once vertex v_t^k is visited, the set of discovered vertices is updated as $V_d = V_d \setminus v_t^k$ and the set of visited vertices is updated as $V_{vis} = V_{vis} \cup v_t^k$.

With multiple robots exploring the environment, it is possible that the robots take different amounts of time to reach their assigned target vertices. If a robot reaches its target vertex earlier than the other robots, the exploration algorithm has three options: (1) the robot waits till the other robots have reached their target vertices; (2) all other robots preemptively stop moving to their assigned target vertices, the target vertices are recomputed, and then exploration continues; or (3) the robot that reaches early is assigned its next target vertex. In both the first and second options, all robots are simultaneously assigned target vertices. The third option does not utilize the updated map, and robots en route to their target vertices do not utilize the updated exploration graph G before reaching their target vertices.

We present two multi-robot exploration algorithms, a priority-based greedy exploration formulation and a multi-robot Profitable Tour Problem based formulation.

4.3.1 Priority-based Greedy Formulation

The objective of the Priority-based Greedy (P-Greedy) algorithm is to greedily assign the highest priority nearby vertex to each robot. This algorithm is a multi-robot extension of the single-robot priority-based greedy algorithm presented in our previous work [41]. In the single-robot priority-based greedy exploration, a single robot was assigned the vertex with the highest priority in the graph. If there were multiple such vertices, the algorithm used the path cost from the current vertex of the robot to the the target vertex as the first tie-breaker and the path cost from the target vertex to the home vertex the second tie-breaker. The second tie-breaker was applicable only when a deadline was imposed during the exploration. The multi-robot extension of the single robot priority-based greedy exploration algorithm considers the all robots and discovered vertices to identify which robot is best greedily assigned to a certain target vertex.

The multi-robot priority-based greedy algorithm loops through all discovered vertices and all unassigned robots to create a list of feasible assignment combinations. These assignment combinations are used to create a max heap with a custom heapify function which maximizes over the priority of the target vertices. This heapify function uses the time estimate $cost_{c_k,d}$ to travel from the robot's current vertex v_c^k to the discovered vertex v_d as a tie-breaker, where a smaller value of the $cost_{c_k,d}$ is preferred. When a deadline is imposed on the robots, the priority function uses the time estimate from the potential target vertex v_d to home v_h as an additional tie-breaker, where a smaller value of $cost_{d,h}$ is preferred. If tie-breakers cannot differentiate between equally good vertex choices, a vertex is chosen randomly. Once a discovered vertex is assigned to a robot k, the vertex is marked as the target vertex v_t^k of robot k. The priority of this target vertex v_t^k is reduced to ε, where ε is an infinitesimally small number close to zero. This encourages subsequent robots to not visit the same target vertex. A new heap is created without the assigned robot and using the updated pri-

Algorithm 4: Multi-robot priority-based greedy algorithm

```
Input: starting vertices: $v_0^k$, $k \in K$; deadline: $t_r \leftarrow unknown$ or $\in \mathbb{R}^+$
Result: Explored graph $G$
$V_d \leftarrow \emptyset$ ;                               // discovered and unvisited vertices
$V_{vis} \leftarrow \{v_0^k\}, k \in K$ ;                                     // visited vertices
Add ($v_0^k$, E($v_0^k$)) to $G$ ;   // E($v_i$) is set of edges incident on vertex $i$
$v_h, v_c^k \leftarrow v_0^k \forall k \in K$;
for robot $k$ in $K$ do
    $V_n^k \leftarrow N_G(v_c^k)$ ;                               // vertices adjacent to $v_c^k$
    $V_d \leftarrow V_d \cup V_n^k$;                             // update discovered vertices
while $V_d \neq \emptyset$ do
    $t_r \leftarrow$ Query_Deadline() ;                                  // check deadline
    if $t_r$ is unknown then
        $V_t \leftarrow$ NextVertexWODeadline($V_d, V_c$);
    else if $t_r$ is 0 then
        return $G$
    else
        $V_t \leftarrow$ NextVertexWDeadline($V_d, V_c$, $v_h$, $t_r$);
    $V_n \leftarrow N_G(V_c)$, $V_n \leftarrow V_n \setminus V_{vis}$, $V_d \leftarrow V_d \cup V_n$ ;
    Move $K$ agents to $V_t$ through graph $G$;
    if Preemption then
        Move robots till first robot $k$ reaches $v_t^k$;
        $V_c \leftarrow$ current robot vertices;
        $V_{vis} \leftarrow V_{vis} \cup V_c$, $V_d \leftarrow V_d \setminus V_{vis}$;
        Add ($V_c$, $V_n$, E($V_c$)) to $G$;
    else
        $V_d \leftarrow V_d \setminus V_t$; $V_c \leftarrow V_t$; $V_{vis} \leftarrow V_{vis} \cup V_c$ ;
        Add ($V_c$, $V_n$, E($V_c$)) to $G$;
if $t_r$ is unknown then
    Move agent to $v_h$ through $G$;
return $G$;
```

ority of the target vertex v_t^k to assign the next robot to a target vertex. This process continues till all robots $k \in K$ are assigned a target vertex v_t^k. Algorithm 4 shows the multi-robot priority-based greedy algorithm. The algorithm uses two functions: to assign the target vertices without a deadline, shown in Algorithm 5, and in the presence of a deadline, as shown in Algorithm 6.

Once each of the robots is assigned a vertex, the robots traverse the graph G till one of the robots $k \in K$ arrives at its target vertex v_t^k. The exploration can be stopped preemptively as soon as this occurs, or each robot can wait at its target

Algorithm 5: `NextVertexWODeadline`: Vertex assignment without deadline

```
Input: $V_d$, $V_c$
Result: Next set of vertices to visit
1  $a_n \leftarrow 0$, $V_t \leftarrow$ empty array of size $|K|$;
2  while $a_n < |K|$ do
3      $V_{list} \leftarrow \emptyset$;
4      for $v_d$ in $V_d$ do
5          for $k$ in $K$ do
6              $cost_{c_k,d} \leftarrow dist(v_c^k, v_d)$;
7              add $v_d^k$ to $V_{list}$
8      $v_t^k \leftarrow \arg\max_{v_d^k \in V_{list}} p_d^k$, tiebreakers: $cost_{c_k,d}$;
9      $a_n \leftarrow a_n + 1$, $K \leftarrow K \setminus k$ ;            // remove robot from set
10     $p_t^k \leftarrow \varepsilon$, $V_t[k] = v_t^k$;
11 return $V_t$;
```

vertex till all robots arrive at their assigned target vertices. Preemptive exploration requires additional computation as robots are frequently stopped due to preemption and require new target vertex assignments to be computed on an updated graph G. In exploration with "Wait", the robots sit idle at the target vertex without exploring the environment. We compare both the variants to identify which one works better for the priority-based greedy solution.

The priority-based greedy algorithm makes the assignment based on the best single vertex it can visit in a one-step look ahead. The greedy algorithm is suitable for robot exploration as we have little information about the initially unknown environment. However, in our single robot prioritized exploration, we have found that non-greedy techniques such as Profitable Tour Problem based exploration can be competitive.

4.3.2 Multi-robot Profitable Tour Problem

The multi-robot Profitable Tour Problem (MR-PTP) is a multi-robot variant of the Profitable Tour Problem. It is a combinatorial optimization problem where the goal is to generate paths that maximize the sum of the profits of the robots from vertex s to vertex t in a graph G, where s and t can be different vertices. A robot's profit is the sum of the prizes collected along its path within the deadline t_r less its path cost. The problem can also be stated as a variation of the Team Orienteering

Algorithm 6: `NextVertexWDeadline`: Vertex assignment with deadline

```
Input: $V_d$, $V_c$, $v_h$, $t_r$
Result: Next set of vertices to visit
1  $a_n \leftarrow 0$, $V_t \leftarrow$ empty array of size $|K|$;
2  while $a_n < |K|$ do
3      $V_{list} \leftarrow \emptyset$;
4      for $v_d$ in $V_d$ do
5          for $k$ in $K$ do
6              if $cost_{c_k,d} + cost_{d,h} > t_r$ then
7                  continue
8              $cost_{c_k,d} \leftarrow dist(v_c^k, v_d)$;
9              add $v_d^k$ to $V_{list}$
10     $v_t^k \leftarrow \arg\max_{v_d^k \in V_{list}} p_d^k$, tiebreakers: $cost_{c_k,d}, cost_{d,h}$;
11     $a_n \leftarrow a_n + 1$, $K \leftarrow K \setminus k$ ;                // remove robot from set
12     $p_t^k \leftarrow \varepsilon$, $V_t[k]= v_t^k$;
13 return $V_t$;
```

Algorithm 7: `Explore_PTP`: Approximation algorithm for multi-robot Profitable Tour Problem

```
Input: $G$, $V_c$, $v_h$, $t_r$, $K$
Result: Set of paths $\mathcal{P}$
1  $\mathcal{P} = \{\}$ ;                                  // initialize set of paths
2  $k \leftarrow 0$ ;                                     // number of paths calculated
3  while $k < |K|$ do
4      $P_k \leftarrow$ RG_PTP$(v_c^k, v_h, V_d, t_r, \{v_c^k, v_h\}, 4)$;
5      $G \leftarrow$ Update_Prizes$(G, P_k)$, $\mathcal{P} \leftarrow \mathcal{P} \cup P_k$, $k \leftarrow k + 1$;
6  return $\mathcal{P}$;
```

Problem [103, 104] which has been used in robot coverage problems [105, 106] and exploration problems [107]. As a solution to MR-PTP calculates a path P_k for each robot $k \in K$, we term it as multi-step lookahead. We expect the robots would have longer than a one-step look ahead of the greedy algorithm and would plan the trajectories efficiently given the limited knowledge of the environment.

In the Team Orienteering Problem (TOP), we assume that we have a complete graph $G = \{V, E\}$, where $V = \{0, \ldots, n\}$. Each vertex $v_i \in V$ has a non-negative prize p_i. The objective of the Team Orienteering Problem is the sum of all prizes collected by the robots from their tour in the graph [103]. The objective is constrained by the time budget or deadline within which the robots have to return to the home

Algorithm 8: `RG_PTP`: Recursive greedy approximation algorithm for the single-robot Profitable Tour Problem

```
    Input: $v_c$, $v_h$, $V_d$, $t_r$, $R$, $iter$
    Result: A path $P$
 1  if $cost(v_c, v_h) > t_r$ then return $Infeasible$;
 2  $P = (v_c, v_h)$ ;                                        // initialize path
 3  if $iter == 0$ then return $P$;
 4  $obj = m.Path\_prize(P) - Path\_cost(P)$;
 5  for $v_d$ in $V_d$ do
 6      for $1 \leq B_1 \leq t_r$ do
 7          $P_1 =$ RG_PTP($v_c$, $v_d$, $V_d$, $B_1$, $R \cup v_d$, $iter - 1$);
 8          if $P_1 == Infeasible$ then continue;
 9          $P_2 =$ RG_PTP($v_d$, $v_h$, $V_d$, $t_r - B_1$, $R \cup P_1$, $iter - 1$);
10          if $P_2 == Infeasible$ then continue;
11          $P_{new} = Concatenate(P_1, P_2)$;
12          $obj_{new} = m.Path\_prize(P_{new}) - Path\_cost(P_{new})$;
13          if $obj_{new} > obj$ then $obj = obj_{new}$, $P = P_{new}$ ;
14  return $P$;
```

position. The Multi-Robot Profitable Tour Problem (MR-PTP) is a variant of the team orienteering problem. Its objective is to maximize, across all robots, the sum of the differences between the prizes collected along each robot's path and the cost of that path, all within the deadline t_r.

MR-PTP is an NP-hard problem [94], with optimal solutions requiring exponential time. For a single robot case, PTP was usable for exploration problems if the number of vertices in the graph was reduced to 15 vertices [108]. To use the MR-PTP for exploration, we have adapted the approximation for the Generalized Team Orienteering Problem by Xu et. al. [109].

Algorithm 7 shows the approximation algorithm for the multi-robot PTP problem. In this algorithm, we initialize a set of empty paths $\mathcal{P}$ for all of the robots. The input to the algorithm is the exploration graph G, the current positions of the K robots V_c, the home vertex v_h, and the deadline t_r by which the robots needs to return home. The algorithm iterates over each of the robots and calculates a path for the robot with the approximate single-robot PTP algorithm shown in Algorithm 8. The path of each robot k starts at its current vertex v_c^k and ends at the home vertex v_h, and

Algorithm 9: `Update_Prizes` Update prizes of graph vertices

```
  Input: $G$, $P$
  Result: $G$
1 for $0 \leq i \leq |P|$ do
2     $v_i \leftarrow i^{\text{th}}$ vertex in path $P$;
3     Prize($v_i$) = $|(1 - 2^{-(i-1)})|$Prize($v_i$);
4 Return graph $G$;
```

must be completed within the deadline t_r. Once a path P_k for robot k has been determined, the prizes of the vertices in the path P_k are discounted to encourage the subsequent robots to visit other vertices rather than the visiting the same vertices in P_k. The vertex prize update function is shown in Algorithm 9.

Algorithm 8 is a recursive greedy approximation algorithm for the single-robot Profitable Tour Problem. The algorithm is a variant of the approximation algorithm for the single vehicle orienteering problem shown by Chekuri and Pal [110] and Singh et. al. [111]. The input to this algorithm is the current vertex of the robot v_c, the home vertex v_h, the set of discovered vertices V_d, the deadline t_r, a set of vertices R to not consider in the path P, initialized to $\{v_c, v_h\}$, and the number of iterations *iter*. The algorithm returns *Infeasible* if the path cost from v_c to v_h is greater than the deadline. The variable *iter* denotes maximum allowed recursion depth of the algorithm. We have used $iter = 4$, as values greater than that negatively impacted the runtime of the solution. The objective of the Profitable Tour Problem is the total path prize, scaled by a multiplier m, less the total path cost. We use a fixed value of 100 for m to ensure that all vertices are visited if within the deadline. We use a larger value of m than for the single robot exploration to maintain a higher value of prizes even with multiple robots. The algorithm loops through the set of discovered vertices V_d and creates splits B_1 of the available deadline t_r. Since choosing a linear set of splits $B_1 \in \{0, 1, 2, \ldots, t_r - 1, t_r\}$ leads to a large amount of computation effort, we have instead used exponential splits $B_1 \in \{2^1, 2^2, 2^3, \ldots, 2^{\log_2 t_r}\}$. When using MR-PTP as an exploration algorithm, once the set of paths $\mathcal{P}$ is calculated, the first vertex of each path $P_k \in \mathcal{P}$ is assigned as the target vertex v_t^k for robot k.

4.4 Experiments

Simulation experiments were conducted on a suite of six simulated environments, which were abstracted as graphs and are depicted in Figures 2.4 and 3.5. These graph environments were used to evaluate the performance of the algorithms without the mapping, localization, and robot navigation components influencing the evaluation. In these experiments, vertex priorities are pre-assigned and do not require additional computational overhead. The experiments were run on an Intel Core i7 9800X with 64 GB of RAM, using the Python (v3.6.8) wrapper of Gurobi Optimizer (v9.5.1).

Further, additional experiments were executed the Gazebo Simulator to assess the practical behavior of the Priority-based Greedy algorithm in a real-time simulated environment. This test involved two robots navigating a Branched Corridor environment, illustrated in Figure 2.5(c). This setup was designed to demonstrate the algorithm's performance along with noise from SLAM and navigation.

4.4.1 Metrics

The performance of the exploration algorithms in the graph environments is evaluated using the percentage of the environment explored as an indicator of the connectivity of the environment. For these graph environments, we compare the performance of (1) Multi-robot priority-based greedy algorithm (P-Greedy), (2) Multi-robot PTP based exploration algorithm (MR-PTP), and (3) Multi-robot cost-based greedy algorithm (C-Greedy) motivated by [5]. The C-Greedy algorithm aims to explore the environment without considering priority values of the vertices and selects vertices based on the path cost. Similar to the P-Greedy algorithm, once a robot is assigned a target vertex, the priority of the vertex is changed to ε encouraging subsequent robots to explore other vertices. The explored regions of the map are represented by the set of explored vertices $V \in G$, where $V = (V_d \cup V_{vis})$. The performance metric is the percentage of explored vertices in the complete environment for a given deadline.

Table 4.1: C-Greedy is cost-based greedy algorithm, P-Greedy is the priority-based greedy algorithm, and MR-PTP is Multi-Robot Profitable Tour Problem.

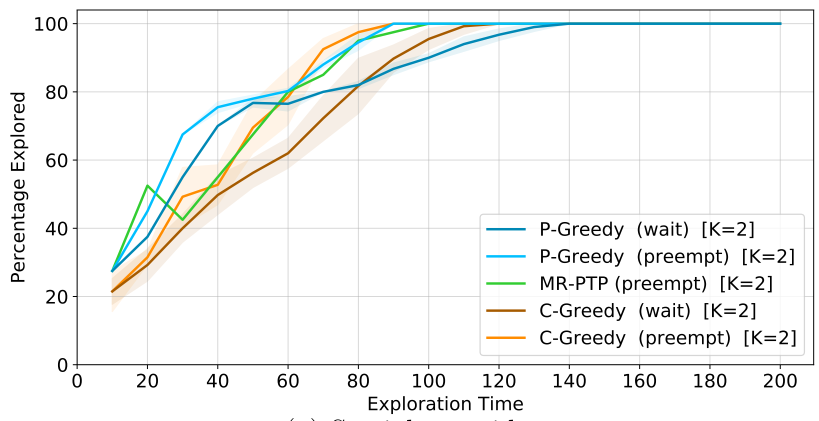
Deadline (t_r) (time steps)	**Algorithm** ($K = 2$)	Corridor	Large Room	Small Room	Total
20	C-Greedy	41.4%	48.8%	17.7%	30.5%
	MR-PTP	**63.6%**	50.0%	30.0%	**45.0%**
	P-Greedy	56.8%	41.2%	28.3%	40.5%
40	C-Greedy	72.7%	75.0%	40.0%	56.7%
	MR-PTP	86.4%	87.5%	60.0%	73.3%
	P-Greedy	**99.1%**	100.0%	62.7%	**81.0%**
80	C-Greedy	98.2%	100.0%	80.0%	89.3%
	MR-PTP	100.0%	100.0%	80.0%	90.0%
	P-Greedy	100.0%	100.0%	78.7%	89.3%

4.4.2 Results and Insights

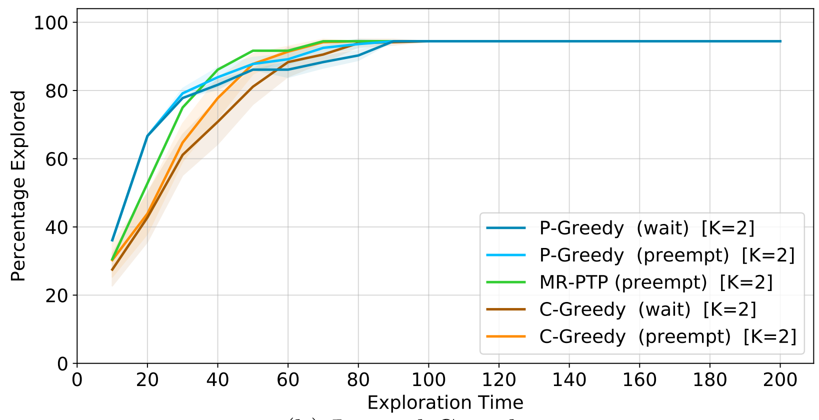
We have compared the results from all six environments as line graphs in Figures 4.3 and 4.4. Here, the X-axis plots the different deadlines within which which the environment is explored, and the Y-axis is the percentage of the environment explored, averaged over ten trials. The shaded region around a line shows the standard deviation of the trials. All ten trials have the same start location.

The MR-PTP exploration algorithm generates paths for each robot to maximize the objective while ensuring the last vertex in the path is the home vertex and the path cost is smaller than the deadline. Similar to single-robot PTP paths [108], a path from MR-PTP does not guarantee that the first vertex the robot visits will be a high priority vertex. This impacts the performance of MR-PTP significantly and in most cases, it is outperformed by the P-Greedy algorithm.

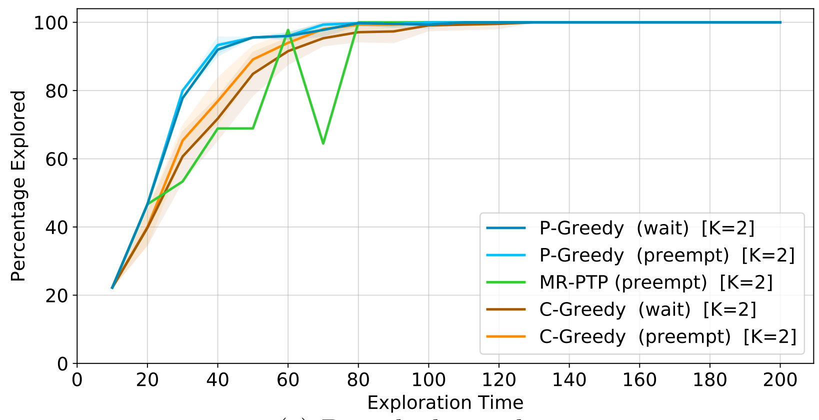
The P-Greedy algorithm, in contrast, chooses the highest priority vertex that is closest to each robot's current position as the robot's target vertex, provided the robot can explore the vertex and return home within the deadline. The P-Greedy algorithm has the best performance for most deadline instances, as shown in Figures 4.3 and 4.4. The only instances where P-Greedy is outperformed by other algorithms by



(a) Straight corridor



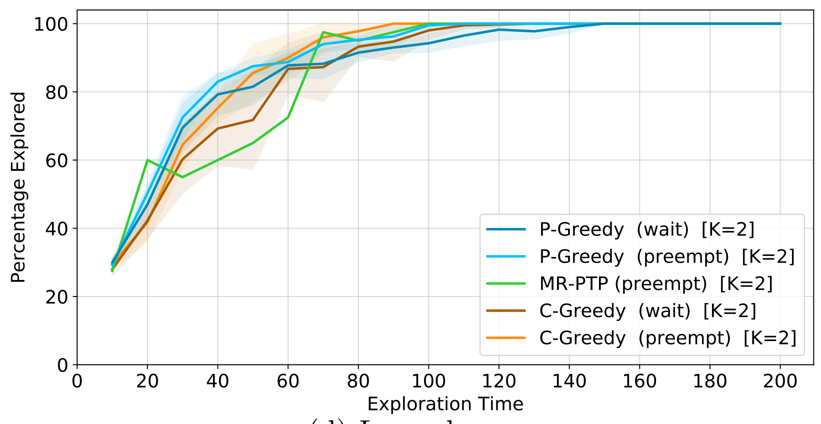
(b) Looped Corridor



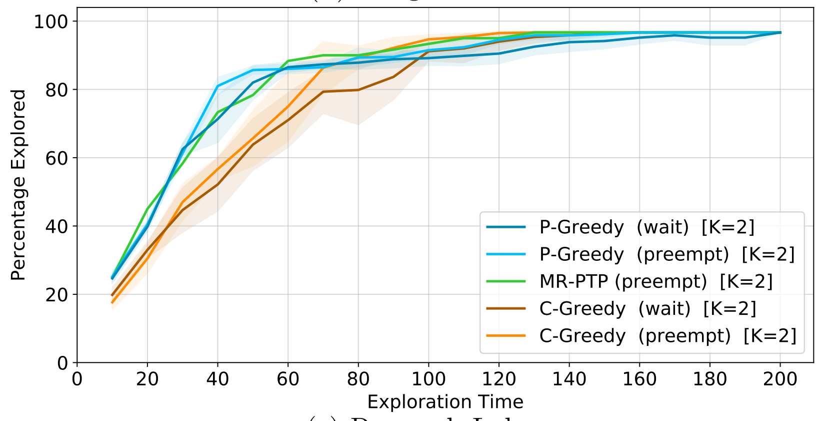
(c) Branched corridor

Figure 4.3: Plots comparing the exploration performance of a team of two robots using three exploration algorithms, P-Greedy, MR-PTP, and C-Greedy for the graph environments of Figure 2.4. The vertical axis shows the average percentage of explored vertices over ten independent runs for each deadline. The shaded regions show the standard deviation of the samples. The horizontal axis shows the deadlines.

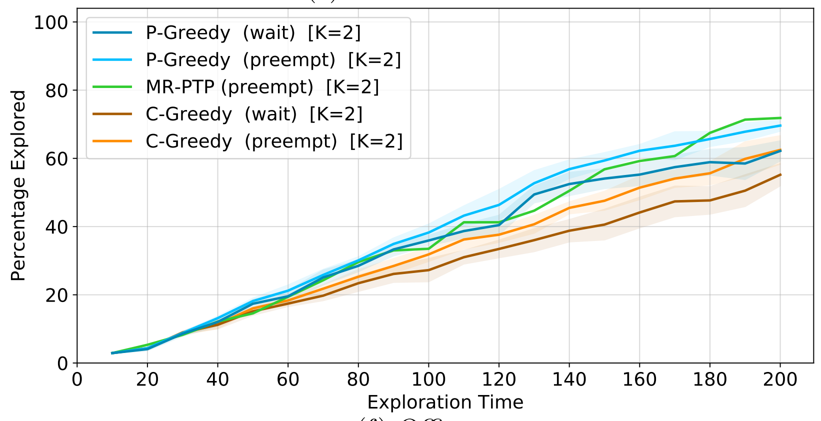
a small margin is for deadlines where over 80 percent of the environment is explored. A few such examples are in the Research Lab environment with exploration time



(d) Large home



(e) Research Lab



(f) Office

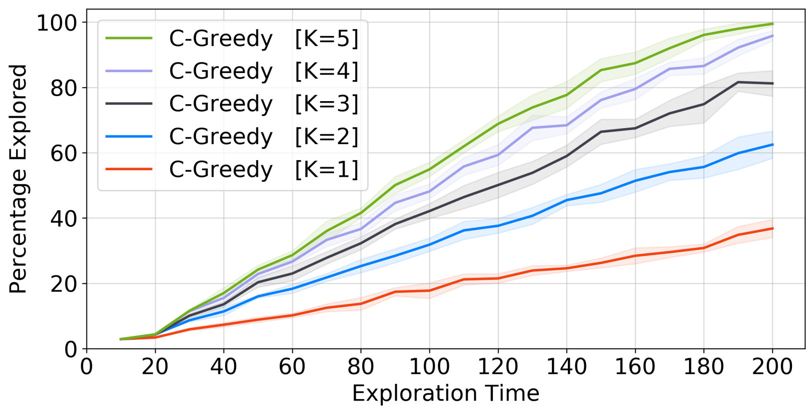
Figure 4.4: Plots comparing the exploration performance of a team of two robots using three exploration algorithms, P-Greedy, MR-PTP, and C-Greedy for the graph environments of Figure 3.5. The vertical axis shows the average percentage of explored vertices over ten independent runs for each deadline. The shaded regions show the standard deviation of the samples. The horizontal axis shows the deadlines. Note that the Office environment shows lower exploration as it requires a larger time to explore due to its larger size. The Looped Corridor and Research Lab have vertices not accessible by any exploration algorithm, keeping their percent explored to less than 100%.

of 110 timesteps or in the Looped Corridor environment with exploration time of 50 timesteps. A consistently strong performance for smaller deadlines makes the P-Greedy algorithm a suitable candidate for deadline driven exploration where the deadline is dynamic.

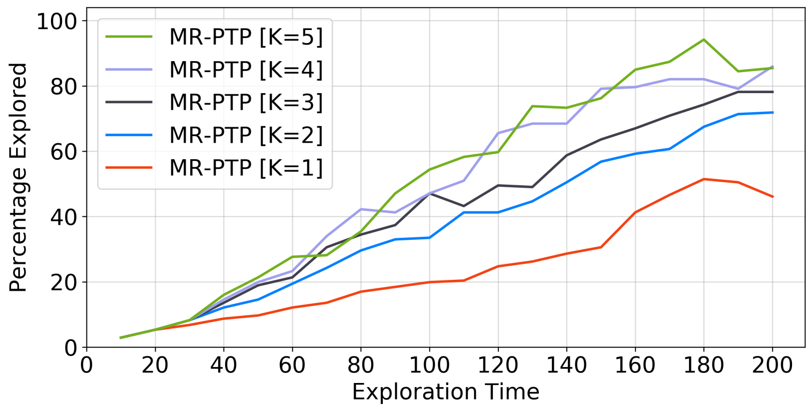
From Figures 4.3 and 4.4, we observe that P-Greedy with preemption always performs better than P-Greedy with wait. We also observe the same behavior with C-Greedy, where C-Greedy with preemption consistently outperforms C-Greedy with wait. Waiting for the other robots without exploring the environment negatively affects the exploration performance.

Table 4.1 compares the performance of the C-Greedy, MR-PTP, and P-Greedy algorithms on the Research Lab environment. The table shows the percentage of the total environment explored and the percentage explored for each building structure: Corridor, Large Room, and Small Room, for three deadlines. Notice that MR-PTP has the best performance at the shortest deadline of $t_r = 20$; however P-Greedy outperforms all other algorithms at $t_r = 40$, while exploring almost the entire corridor.

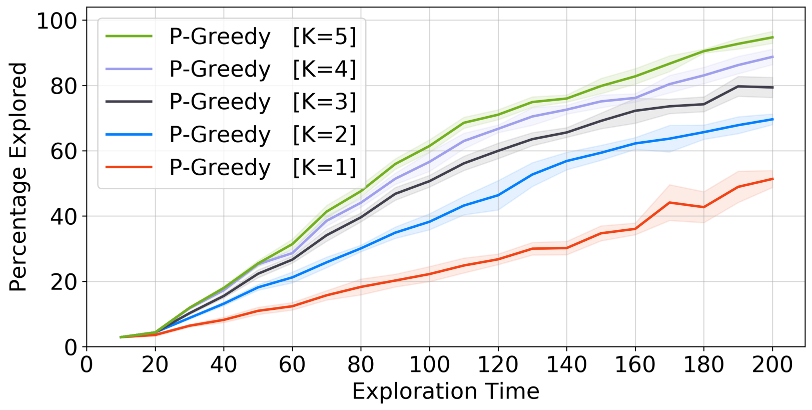
Figure 4.5 compares the performance of exploration with 1 to 5 robots in the Office environment. We choose this environment as it is the largest and clearly shows the differences when using different numbers of robots. With a larger number of robots, $K = 5$, observe that C-Greedy and P-Greedy perform similarly for a small deadline ($t_r < 40$). In such a case, for each step of the exploration, once a high priority vertex is assigned to a robot, the other robots choose low priority vertices. This makes the robots visit vertices with lower connectivity frequently, affecting the performance. When the deadline is about 100 time steps, we notice that P-Greedy performs much better than C-Greedy. By this deadline, each of the robots has identified a corridor for itself, making the P-Greedy exploration more efficient. For a long deadline (e.g., $t_r > 170$), only a few sparsely distributed low priority vertices remain towards the end of the exploration; the robots require more time to visit them, casing the performance



(a) Cost-based greedy



(b) Multi-Robot Profitable Tour Problem



(c) Priority-based greedy

Figure 4.5: Performance comparison of using one to five robots in the Office environment. All algorithm performances improve as the number of robots increases. The cost-based greedy algorithm shows the highest performance benefit with an increasing number of robots.

of P-Greedy to taper off.

Multi-robot Gazebo experiments: In the Branched Corridor Gazebo environment (Figure 2.5(c)), experiments were conducted using two Turtlebot3 Burger

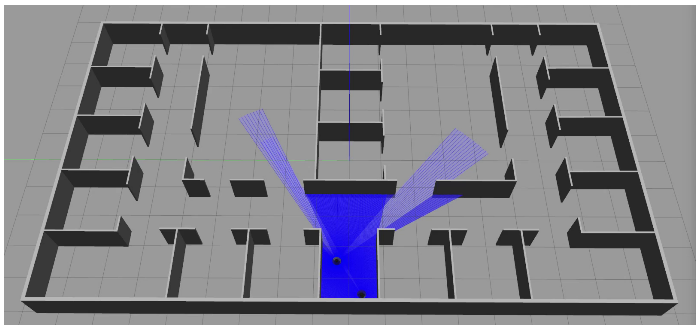
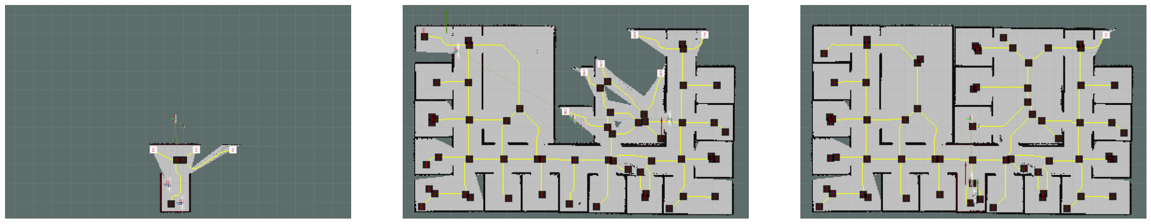


Figure 4.6: Branched Corridor environment with two turtlebot3 robots. The blue lines show the LiDAR simulation of each of the robots. The saturated blue lines show the LiDAR sensor data that has been used for mapping and the other blue lines show the LiDAR rays that do not reach an obstacle.



(a) (b) (c)

Figure 4.7: Exploration of the Branched Corridor Gazebo environment by two Turtlebot3 robots using the Priority-based Greedy algorithm. The three figures show the map generated by two robots at different timestamps of exploration. **(a)** shows the map at the start of exploration. **(b)** shows the map at 500 seconds, and **(c)** shows the map at 1000 seconds. Note that the robots could not complete the exploration in the top right corner due to the deadline imposed on them.

robots [112], as illustrated in Figure 4.6. These robots have a deadline of 500 seconds imposed after 500 seconds of exploration, making the total exploration time 1000 seconds. The robots follow the Priority-based Greedy algorithm with preemption. Equipped with a LiDAR with a range of 6 meters, the robots map the environment individually using the GMapping SLAM algorithm [25]. To consolidate the map data calculated by each robot, a multi-robot map merging technique [113, 114] was utilized, integrating each robot's map into a single occupancy grid map. The robots use the ROS Navigation package based on Dynamic Window Approach (DWA) [26] to navigate around obstacles and around each other.

We illustrate the progress of exploration by showing the merged map at differ-

Table 4.2: Exploration results for the Branched Corridor Gazebo environment as shown in Figure 4.6 with two robots. P-Greedy is the prioritized-greedy algorithm, MR-PTP is the multi-robot PTP-based exploration algorithm with multiplier $m = 100$.

Deadline (in secs.)	Algorithm	Corridor	Large Room	Small Room	Total
500	P-Greedy [K=2]	84.0%	67.9%	40.6%	56.5%
	MR-PTP [K=2]	67.6%	59.2%	27.2%	43.6%
100	P-Greedy [K=2]	95.1%	100.0%	55.4%	75.2%
	MR-PTP [K=2]	91.2%	93.5%	53.1%	71.0%

ent instants during the exploration in the Figure 4.7. Notice that the robot picks two independent corridors exploring different parts of the environment as shown in Figure 4.7(b). The robots can complete the exploration and return back to the starting location before deadline. Here, the robots could explore 78% of the whole environment.

In this Gazebo simulation, we compared the performance of the short-horizon Priority-based Greedy (P-Greedy) algorithm against the long-horizon Multi-Robot Path Planning (MR-PTP) algorithm. For these simulation experiments, the deadline was imposed at the start of exploration. The results are presented in Table 4.2. Observe that the short-horizon P-Greedy algorithm consistently outperforms the long-horizon MR-PTP algorithm across all deadlines. Furthermore, the P-Greedy algorithm was more effective in explore high connectivity areas such as corridors and large rooms, compared to the MR-PTP.

4.5 Contributions

In this chapter, we presented two formulations for the multi-robot prioritized exploration problem: as a short-horizon Priority-based Greedy algorithm and as a long-horizon multi-robot Profitable Tour Problem. Building on an approximation algorithm for the Team Orienteering Problem, we developed a heuristic solution for the multi-robot Profitable Tour Problem.

4.6 Conclusion

This chapter presents multi-robot exploration algorithms for the problem of prioritized exploration with dynamic deadlines. The goal is to rapidly determine the geometric structure and connectivity of the environment by a team of robots. The indoor environment is modeled as a graph, and the robots determine the next target vertex to visit. We have compared three multi-robot exploration algorithms: cost-based greedy, priority-based greedy, and the multi-robot Profitable Tour Problem based algorithm. In all, we preemptively stop the exploration as soon as one of the robots reaches its target vertex. To keep the computation time low, our multi-robot Profitable Tour Problem based exploration algorithm is an adaptation of a multi-robot Orienteering approximation algorithm [109]. We observe a significant speedup in exploration when using multiple robots instead of a single robot, demonstrating that a multi-robot implementation of our algorithm is effective in exploring unknown environments.

We have identified several directions for future work. First, make the P-Greedy algorithm more efficient. Second, implement the algorithms on a team of physical robots, which requires developing robust solutions for map merging. Third, use neural network based classifiers to semantically classify physical locations by their building structures (e.g., corridor).

CHAPTER 5: CONCLUSION

This dissertation addresses the prioritized exploration problem for both single-robot and multi-robot systems, with the objective of rapidly computing the geometric layout of an initially unknown environment. The exploration algorithms introduced in this dissertation enable efficient exploration and ensure the robots can return to the home location within a specified deadline.

For the single-robot prioritized exploration problem, we developed the short-horizon Priority-based Greedy algorithm and explored long-horizon exploration algorithms based on the Orienteering Problem, Profitable Tour Problem, Minimum Latency Paths Problem, and Profitable Tour Problem with Minimum Latency Paths (Chapter 2 and Chapter 3). The Priority-based Greedy algorithm is a one-step lookahead algorithm that identifies the highest priority vertex closest to the robot and directs the robot towards it. Modeling the single-robot prioritized exploration problem as an Orienteering Problem or Profitable Tour Problem allows for a multi-step lookahead. Since both these formulations consider the total prize collected over the collected path, they can cause the robot to visit less connected vertices earlier in the computed path. This leads to reduced exploration performance. The Minimum Latency Paths Problem and Profitable Tour Problem with Minimum Latency Paths formulations allow for a certain number of lookahead steps. The priority of a vertex is determined by the building structure in which it is located. The Priority-based Greedy algorithm moves the robot to the closest highest-priority vertices, making this algorithm particularly effective in short deadlines and outperforming other prioritized exploration algorithms most of the time. Additionally, it operates much faster than other algorithms, with computation time being several orders of magnitude faster. However, the performance of the Priority-based Greedy algorithm exhibits high variability. While it provides rapid exploration, for more consistent performance guarantees, opting for long-horizon prioritized exploration algorithms may prove advantageous. We determine that for the single-robot prioritized exploration problem, the priority-based greedy algorithm is the most effective in most instances.

For the multi-robot prioritized exploration problem, we formulated it as the Multi-robot Priority-based Greedy Problem and Multi-robot Profitable Tour Problem (Chapter 4). To keep the computation time low, our multi-robot Profitable Tour Problem based exploration algorithm is an adaptation of a team orienteering approximation algorithm [109]. We observe a significant speedup in exploration when using multiple robots instead of a single robot, establishing that a multi-robot implementation of our algorithm is effective in exploring unknown environments. In the graph environments, we found that a Priority-based Greedy exploration algorithm performs on par or better than the optimization based algorithms in most instances. In the Gazebo environments, we found that the Priority-based Greedy algorithm outperformed the multi-robot Profitable Tour Problem based exploration algorithm.

There are several directions for future work that we have identified. First, it is essential to conduct additional testing of the prioritized exploration algorithms in real-world environments to gain a deeper understanding of their performance. This includes testing across a broader range of environments. Second, addressing computation and network delays in real-world experiments so that the robot can calculate quicker map updates than the current SLAM algorithm. Third, the implemented multi-robot exploration technique does not account for dynamic obstacles; implementing control barrier functions could enhance safety by preventing robot-robot collisions. Fourth, deriving the values of vertex priorities based on the semantic information provided by the building structures could enable context-dependent prioritization of the vertices. Finally, developing new metrics for prioritized exploration that focus on the layout and connectivity of the explored area to evaluate exploration performance would provide a more nuanced assessment of exploration efficiency.

REFERENCES

[1] B. Yamauchi, “A frontier-based approach for autonomous exploration.” in *Computational Intelligence in Robotics and Automation*, vol. 97, 1997, pp. 146–151.

[2] C. Stachniss, Ó. M. Mozos, and W. Burgard, “Efficient exploration of unknown indoor environments using a team of mobile robots,” *Annals of Mathematics and Artificial Intelligence*, vol. 52, no. 2-4, pp. 205–227, 2008.

[3] C. Wang, D. Zhu, T. Li, M. Q.-H. Meng, and C. W. de Silva, “Efficient autonomous robotic exploration with semantic road map in indoor environments,” *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2989–2996, 2019.

[4] M. T. Ohradzansky, A. B. Mills, E. R. Rush, D. G. Riley, E. W. Frew, and J. S. Humbert, “Reactive control and metric-topological planning for exploration,” in *IEEE International Conference on Robotics and Automation*, 2020, pp. 4073–4079.

[5] B. Yamauchi, “Frontier-based exploration using multiple robots,” in *International Conference on Autonomous Agents*, vol. 98, 1998, pp. 47–53.

[6] C. Stachniss, G. Grisetti, and W. Burgard, “Information gain-based exploration using Rao-Blackwellized particle filters,” in *Robotics: Science and Systems*, vol. 2, 2005, pp. 65–72.

[7] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics.* MIT Press, Sep. 2005.

[8] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, “Robotic exploration as graph construction,” *IEEE Transactions on Robotics and Automation*, vol. 7, no. 6, pp. 859–865, Dec. 1991.

[9] B. Kuipers and Y.-T. Byun, “A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations,” *Journal of Robotics and Autonomous Systems*, vol. 8, no. 1/2, pp. 47–63, 1993.

[10] A. Cowley, C. J. Taylor, and B. Southall, “Rapid multi-robot exploration with topometric maps,” in *IEEE International Conference on Robotics and Automation*, 2011, pp. 1044–1049.

[11] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter, “Graph-based subterranean exploration path planning using aerial and legged robots,” *Journal of Field Robotics*, vol. 37, no. 8, pp. 1363–1388, 2020.

[12] J. Bayer, P. Čížek, and J. Faigl, “Autonomous multi-robot exploration with ground vehicles in DARPA subterranean challenge finals,” *Field Robotics*, vol. 3, no. 1, pp. 266–300, January 2023.

[13] O. Peltzer, A. Bouman, S.-K. Kim, R. Senanayake, J. Ott, H. Delecki, M. Sobue, M. J. Kochenderfer, M. Schwager, J. Burdick, and A.-a. Agha-mohammadi, “Fig-op: Exploring large-scale unknown environments on a fixed time budget,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 8754–8761.

[14] Y. Li, A. Debnath, G. J. Stein, and J. Košecká, “Learning-Augmented Model-Based planning for visual exploration,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023, pp. 5165–5171.

[15] K. Cesare, R. Skeele, S.-H. Yoo, Y. Zhang, and G. Hollinger, “Multi-UAV exploration with limited communication and battery,” in *IEEE International Conference on Robotics and Automation*, 2015, pp. 2230–2235.

[16] S. Oßwald, M. Bennewitz, W. Burgard, and C. Stachniss, “Speeding-up robot exploration by exploiting background information,” *IEEE Robotics and Automation Letters*, vol. 1, no. 2, pp. 716–723, 2016.

[17] A. J. Smith and G. A. Hollinger, “Distributed inference-based multi-robot exploration,” *Autonomous Robots*, vol. 42, no. 8, pp. 1651–1668, 2018.

[18] S. Moon, O. Peltzer, J. Ott, S.-K. Kim, and A.-A. Agha-Mohammadi, “Semantics-aware mission adaptation for autonomous exploration in urban environments,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023, pp. 2065–2070.

[19] D. Zhu, T. Li, D. Ho, C. Wang, and M. Q.-H. Meng, “Deep reinforcement learning supervised autonomous exploration in office environments,” in *IEEE International Conference on Robotics and Automation*, 2018, pp. 7548–7555.

[20] O. Zhelo, J. Zhang, L. Tai, M. Liu, and W. Burgard, “Curiosity-driven exploration for mapless navigation with deep reinforcement learning,” *arXiv preprint arXiv:1804.00456*, 2018.

[21] H. Moravec and A. Elfes, “High resolution maps from wide angle sonar,” in *IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 116–121.

[22] T. Zhang and C. Y. Suen, “A fast parallel algorithm for thinning digital patterns,” *Communications of the ACM*, vol. 27, pp. 236–239, 1984.

[23] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “OctoMap: An efficient probabilistic 3D mapping framework based on octrees,” *Autonomous Robots*, 2013. [Online]. Available: http://octomap.github.com

[24] C. E. Aguero, N. Koenig, I. Chen, H. Boyer, S. Peters, J. Hsu, B. Gerkey, S. Paepcke, J. L. Rivero, J. Manzo, E. Krotkov, and G. Pratt, “Inside the Virtual Robotics Challenge: Simulating Real-Time Robotic Disaster Response,” *IEEE*

Transactions on Automation Science and Engineering, vol. 12, pp. 494–506, April 2015.

[25] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[26] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics and Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[27] K. Zheng, "Ros navigation tuning guide," 2019.

[28] H. Choset, "Coverage for robotics–a survey of recent results," *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 113–126, 2001.

[29] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, 2013.

[30] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of Computer Computations.* Springer, 1972, pp. 85–103.

[31] C. Connolly, "The determination of next best views," in *IEEE International Conference on Robotics and Automation*, vol. 2, 1985, pp. 432–435.

[32] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, "Receding horizon "next-best-view" planner for 3D exploration," in *IEEE International Conference on Robotics and Automation*, 2016, pp. 1462–1468.

[33] Y. Cai, S. X. Yang, and X. Xu, "A combined hierarchical reinforcement learning based approach for multi-robot cooperative target searching in complex unknown environments," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, 2013, pp. 52–59.

[34] C. Robin and S. Lacroix, "Multi-robot target detection and tracking: taxonomy and survey," *Autonomous Robots*, vol. 40, no. 4, pp. 729–760, 2016.

[35] W.-C. Lee and H.-L. Choi, "Complex semantic-spatial relation aided indoor target-directed exploration," *IEEE Access*, vol. 9, pp. 167 039–167 053, 2021.

[36] D. G. Vutetakis and J. Xiao, "An autonomous loop-closure approach for simultaneous exploration and coverage of unknown infrastructure using MAVs," in *IEEE International Conference on Robotics and Automation*, 2019, pp. 2988–2994.

[37] I. Lluvia, E. Lazkano, and A. Ansuategi, "Active mapping and robot exploration: A survey," *Sensors*, vol. 21, no. 7, 2445, 2021.

[38] D. Pittol, M. Mantelli, R. Maffei, M. Kolberg, and E. Prestes, "Loop-aware exploration graph: A concise representation of environments for exploration and active loop-closure," *Robotics and Autonomous Systems*, vol. 155, p. 104179, 2022.

[39] T. Sakamoto, S. Bonardi, and T. Kubota, "A routing framework for heterogeneous multi-robot teams in exploration tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6662–6669, 2020.

[40] G. Best and G. A. Hollinger, "Decentralised self-organising maps for the online orienteering problem with neighbourhoods," in *International Symposium on Multi-Robot and Multi-Agent Systems.* IEEE, 2019, pp. 139–141.

[41] S. Datta and S. Akella, "Prioritized indoor exploration with a dynamic deadline," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 3108–3114.

[42] I.-M. Chao, B. L. Golden, and E. A. Wasil, "A fast and effective heuristic for the orienteering problem," *European Journal of Operational Research*, vol. 88, no. 3, pp. 475–489, 1996.

[43] P. Vansteenwegen, W. Souffriau, and D. Van Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[44] B. L. Golden, L. Levy, and R. Vohra, "The orienteering problem," *Naval Research Logistics*, vol. 34, no. 3, pp. 307–318, 1987.

[45] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[46] D. Feillet, P. Dejax, and M. Gendreau, "Traveling salesman problems with profits," *Transportation Science*, vol. 39, no. 2, pp. 188–205, 2005.

[47] M. Fischetti, G. Laporte, and S. Martello, "The delivery man problem and cumulative matroids," *Operations Research*, vol. 41, no. 6, pp. 1055–1064, 1993.

[48] J. Mikula and M. Kulich, "Solving the traveling delivery person problem with limited computational time," *Central European Journal of Operations Research*, vol. 30, no. 4, pp. 1451–1481, 2022.

[49] K. Chaudhuri, B. Godfrey, S. Rao, and K. Talwar, "Paths, trees, and minimum latency tours," in *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* IEEE, 2003, pp. 36–45.

[50] A. M. Campbell, D. Vandenbussche, and W. Hermann, "Routing for relief efforts," *Transportation science*, vol. 42, no. 2, pp. 127–145, 2008.

[51] R. W. Floyd, "Algorithm 97: shortest path," *Communications of the ACM*, vol. 5, no. 6, p. 345, 1962.

[52] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM*, vol. 7, no. 4, pp. 326–329, 1960.

[53] M. M. Silva, A. Subramanian, T. Vidal, and L. S. Ochi, "A simple and effective metaheuristic for the minimum latency problem," *European Journal of Operational Research*, vol. 221, no. 3, pp. 513–520, Sep 2012.

[54] L. Bianco, A. Mingozzi, and S. Ricciardelli, "The traveling salesman problem with cumulative costs," *Networks*, vol. 23, no. 2, pp. 81–91, 1993.

[55] H.-S. Park and C.-H. Jun, "A simple and fast algorithm for k-medoids clustering," *Expert systems with applications*, vol. 36, no. 2, pp. 3336–3341, 2009.

[56] R. M. Karp, "On-line algorithms versus off-line algorithms: How much is it worth to know the future?" in *IFIP Congress*, vol. 1, 1992, pp. 416–429.

[57] A. Borodin and R. El-Yaniv, *Online Computation and Competitive Analysis*. Cambridge University Press, 2005.

[58] L. Iocchi, D. Nardi, and M. Salerno, "Reactivity and deliberation: a survey on multi-robot systems," in *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*. Springer, 2000, pp. 9–32.

[59] N. Bartolini, A. Coletta, G. Maselli, and A. Khalifeh, "A multi-trip task assignment for early target inspection in squads of aerial drones," *IEEE Transactions on Mobile Computing*, vol. 20, no. 11, pp. 3099–3116, 2021.

[60] S. Agarwal and S. Akella, "The single robot line coverage problem: Theory, algorithms, and experiments," *Networks*, vol. 82, no. 4, pp. 479–505, 2023. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.22171

[61] R. Cipolleschi, M. Giusto, A. Q. Li, and F. Amigoni, "Semantically-informed coordinated multirobot exploration of relevant areas in search and rescue settings," in *European Conference on Mobile Robots*, 2013, pp. 216–221.

[62] D. S. Drew, "Multi-agent systems for search and rescue applications," *Current Robotics Reports*, vol. 2, no. 2, pp. 189–200, 2021.

[63] S. Agarwal and S. Akella, "Simultaneous optimization of assignments and goal formations for multiple robots," in *IEEE International Conference on Robotics and Automation*, Brisbane, Australia, May 2018, pp. 6708–6715.

[64] J. Alonso-Mora, E. Montijano, T. Nägeli, O. Hilliges, M. Schwager, and D. Rus, "Distributed multi-robot formation control in dynamic environments," *Autonomous Robots*, vol. 43, no. 5, pp. 1079–1100, 2019.

[65] I. M. Rekleitis, G. Dudek, and E. E. Milios, "Multi-robot exploration of an unknown environment, efficiently reducing the odometry error," in *International Joint Conference on Artificial Intelligence*, vol. 15, 1997, pp. 1340–1345.

[66] C. Stachniss, "Exploration and mapping with mobile robots," Ph.D. dissertation, University of Freiburg, Department of Computer Science, April 2006.

[67] C. Stachniss and W. Burgard, "Exploring unknown environments with mobile robots using coverage maps," in *International Joint Conference on Artificial Intelligence*, vol. 2003, 2003, pp. 1127–1134.

[68] N. Fairfield, G. Kantor, and D. Wettergreen, "Real-time SLAM with octree evidence grids for exploration in underwater tunnels," *Journal of Field Robotics*, vol. 24, no. 1-2, pp. 03–21, 2007.

[69] L. A. Andersson and J. Nygards, "C-SAM: Multi-robot SLAM using square root information smoothing," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 2798–2805.

[70] J. McDonald, M. Kaess, C. Cadena, J. Neira, and J. J. Leonard, "6-DOF multi-session visual SLAM using anchor nodes," in *European Conference on Mobile Robots*, September 2011, pp. 69–76.

[71] S. Datta, A. Sharma, and K. M. Krishna, "Multi-trajectory pose correspondences using scale-dependent topological analysis of pose-graphs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems.* IEEE, 2017, pp. 1019–1025.

[72] K. Konolige, D. Fox, B. Limketkai, J. Ko, and B. Stewart, "Map merging for distributed robot navigation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2003, pp. 212–217.

[73] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, "Collaborative multi-robot exploration," in *IEEE International Conference on Robotics and Automation*, 2000, pp. 476–481.

[74] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, "Coordinated multi-robot exploration," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 376–386, 2005.

[75] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, "Distributed multirobot exploration and mapping," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1325–1339, 2006.

[76] K. M. Wurm, C. Stachniss, and W. Burgard, "Coordinated multi-robot exploration using a segmentation of the environment," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 1160–1165.

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

[78] A. Benkrid, A. Benallegue, and N. Achour, "Multi-robot coordination for energy-efficient exploration," *Journal of Control, Automation and Electrical Systems*, vol. 30, no. 6, pp. 911–920, 2019.

[79] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping," in *AAAI Conference on Artificial Intelligence*, 2000, pp. 852–858.

[80] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *IEEE International Conference on Robotics and Automation*, vol. 3, 2002, pp. 3016–3023.

[81] J. Hawley and Z. Butler, "Hierarchical distributed task allocation for multi-robot exploration," in *Distributed Autonomous Robotic Systems.* Berlin, Heidelberg: Springer, 2013, pp. 445–458.

[82] M. Otte, M. J. Kuhlman, and D. Sofge, "Auctions for multi-robot task allocation in communication limited environments," *Autonomous Robots*, vol. 44, no. 3, pp. 547–584, 2020.

[83] S. M. LaValle, *Planning Algorithms.* Cambridge, U.K.: Cambridge University Press, 2006, available at http://planning.cs.uiuc.edu/.

[84] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

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

[86] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[87] J. Bruce and M. M. Veloso, "Real-time randomized path planning for robot navigation," in *RoboCup 2002: Robot Soccer World Cup VI.* Berlin, Heidelberg: Springer, 2003, pp. 288–295.

[88] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles.* Springer, 1986, pp. 396–404.

[89] A. Stentz, "The focussed D* algorithm for real-time replanning," in *International Joint Conference on Artificial Intelligence*, vol. 95, 1995, pp. 1652–1659.

[90] A. Stentz, "Optimal and efficient path planning for partially known environments," in *Intelligent Unmanned Ground Vehicles.* Springer, 1997, pp. 203–220.

[91] J. Vannoy and J. Xiao, "Real-time adaptive motion planning (RAMP) of mobile manipulators in dynamic environments with unforeseen changes," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1199–1212, 2008.

[92] S. McLeod, "Robust and reliable real-time adaptive motion planning," Ph.D. dissertation, University of North Carolina at Charlotte, 2019.

[93] K. Zhang, S. McLeod, M. Lee, and J. Xiao, "Continuous reinforcement learning to adapt multi-objective optimization online for robot motion," *International Journal of Advanced Robotic Systems*, vol. 17, no. 2, 2020.

[94] 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.

[95] G. Hardouin, J. Moras, F. Morbidi, J. Marzat, and E. M. Mouaddib, "Next-Best-View planning for surface reconstruction of large-scale 3D environments with multiple UAVs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2020, pp. 1567–1574.

[96] L. Freda, T. Novo, D. Portugal, and R. P. Rocha, "3D multi-robot exploration with a two-level coordination strategy and prioritization," *arXiv preprint arXiv:2307.02417*, 2023.

[97] I. Rekleitis, V. Lee-Shue, A. P. New, and H. Choset, "Limited communication, multi-robot team based coverage," in *IEEE International Conference on Robotics and Automation*, vol. 4, 2004, pp. 3462–3468.

[98] M. Kulkarni, M. Dharmadhikari, M. Tranzatto, S. Zimmermann, V. Reijgwart, P. De Petris, H. Nguyen, N. Khedekar, C. Papachristos, L. Ott, R. Siegwart, M. Hutter, and K. Alexis, "Autonomous teamed exploration of subterranean environments using legged and aerial robots," in *International Conference on Robotics and Automation*, 2022, pp. 3306–3313.

[99] A. Bouman, J. Ott, S.-K. Kim, K. Chen, M. J. Kochenderfer, B. Lopez, A.-A. Agha-Mohammadi, and J. Burdick, "Adaptive coverage path planning for efficient exploration of unknown environments," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2022, pp. 11 916–11 923.

[100] J. Faigl and M. Kulich, "On determination of goal candidates in frontier-based multi-robot exploration," in *European Conference on Mobile Robots*. IEEE, 2013, pp. 210–215.

[101] T. Andre and C. Bettstetter, "Collaboration in multi-robot exploration: to meet or not to meet?" *Journal of Intelligent and Robotic Systems*, vol. 82, pp. 325–337, 2016.

[102] V. C. Kalempa, L. Piardi, M. Limeira, and A. S. de Oliveira, “Multi-robot preemptive task scheduling with fault recovery: A novel approach to automatic logistics of smart factories,” *Sensors*, vol. 21, no. 19, p. 6536, 2021.

[103] S. E. Butt and T. M. Cavalier, “A heuristic for the multiple tour maximum collection problem,” *Computers and Operations Research*, vol. 21, no. 1, pp. 101–111, 1994. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0305054894900655

[104] I.-M. Chao, B. L. Golden, and E. A. Wasil, “The team orienteering problem,” *European Journal of Operational Research*, vol. 88, no. 3, pp. 464–474, 1996.

[105] D. Thakur, M. Likhachev, J. Keller, V. Kumar, V. Dobrokhodov, K. Jones, J. Wurz, and I. Kaminer, “Planning for opportunistic surveillance with multiple robots,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 5750–5757.

[106] A. Mansfield, S. Manjanna, D. G. Macharet, and M. A. Hsieh, “Multi-robot scheduling for environmental monitoring as a team orienteering problem,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2021, pp. 6398–6404.

[107] M. Poggi, H. Viana, and E. Uchoa, “The team orienteering problem: Formulations and branch-cut and price,” in *10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS’10)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.

[108] S. Datta and S. Akella, “Prioritized robotic exploration with deadlines: A comparison of greedy, orienteering, and profitable tour approaches,” in *IEEE International Conference on Robotics and Automation*, 2023, pp. 5737–5743.

[109] W. Xu, W. Liang, Z. Xu, J. Peng, D. Peng, T. Liu, X. Jia, and S. K. Das, “Approximation algorithms for the generalized team orienteering problem and its applications,” *IEEE/ACM Transactions on Networking*, vol. 29, no. 1, pp. 176–189, 2020.

[110] C. Chekuri and M. Pal, “A recursive greedy algorithm for walks in directed graphs,” in *IEEE Symposium on Foundations of Computer Science*, 2005, pp. 245–253.

[111] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, “Efficient informative sensing using multiple robots,” *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[112] ROBOTIS, “TurtleBot 3 Burger.” [Online]. Available: https://www.robotis.us/turtlebot-3-burger-us/

[113] J. Hörner, “Map-merging for multi-robot system,” Bachelor’s thesis, Charles University in Prague, Faculty of Mathematics and Physics, Prague, 2016. [Online]. Available: https://is.cuni.cz/webapps/zzp/detail/174125/

[114] K. Smith, “Multi-robot exploration and map merging,” https://github.com/gingineer95/Multi-Robot-Exploration-and-Map-Merging, 2021.