AN ABSTRACT OF THE THESIS OF

Chris (Yu Hsuan) Lee for the degree of Master of Science in Robotics presented on September 16, 2021.

Title: Deployment Planning for Multiple Marsupial Robots

Abstract approved: ______________________________

Geoffrey Hollinger

Multi-robot systems are versatile and extremely capable of exploration tasks in complex environments. Increasingly sophisticated planners, which incorporate new features of a multi-robot system, are necessary for the operation of the systems. Marsupial robots are multi-robot systems consisting of a *carrier* robot (e.g., a ground vehicle), which is highly capable and has a long mission duration, and at least one *passenger* robot (e.g., a short-duration aerial vehicle) transported by the carrier. These heterogeneous robot systems are more flexible than traditional homogeneous multi-robot systems in handling the constraints of a complex environment. However, the physical coupling of a carrier robot and passenger robots requires planners that are capable of accounting for passenger robot deployment planning in exploration. In this thesis, we present two algorithms for deploying passenger robots from marsupial robot systems, culminating into a generalized deployment framework.

First, we address the single marsupial robot case: deploying multiple passenger robots from a single carrier robot. We optimize the performance of passenger robot deployment by using an algorithm that reasons over uncertainty by exploiting information about the prior probability distribution of features of interest in the environment. Our algorithm is formulated as a solution to a sequential stochastic assignment problem (SSAP). The key feature of the algorithm is a recurrence relationship that defines a set of observation thresholds that are used to decide when to deploy passenger robots. Our algorithm computes the optimal policy in $O(NR)$ time, where N is the number of deployment decision points and R is the number of passenger robots to be deployed. We conducted drone deployment exploration experiments on real-world data from the DARPA Subterranean challenge to test the SSAP algorithm. Our results show that our deployment algorithm outperforms other competing algorithms, such as the classic secretary approach and baseline partitioning methods, and is comparable to an offline oracle algorithm.

Secondly, we expand to the multiple marsupial robot case, deploying multiple passenger robots from multiple carrier robots. While the single marsupial robot deployment algorithm is an optimal, polynomial-time solution, the multiple-carrier robot deployment problem is fundamentally harder as it requires addressing conflicts and dependencies between deployments of multiple passenger robots. We propose a centralized heuristic search algorithm for the multiple-carrier robot deployment problem that combines Monte Carlo tree search with a dynamic programming-based solution to the Sequential Stochastic Assignment Problem as a rollout action-selection policy. Our results with both procedurally-generated data and data drawn from the DARPA Subterranean Challenge Urban Circuit show the viability of our approach and substantial exploration performance improvements over alternative algorithms.

Deployment Planning for Multiple Marsupial Robots

by

Chris (Yu Hsuan) Lee

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented September 16, 2021
Commencement June 2022

Master of Science thesis of Chris (Yu Hsuan) Lee presented on September 16, 2021.

APPROVED:

Major Professor, representing Robotics

Associate Dean of Graduate Studies, College of Engineering

Dean of the Graduate School

Chris (Yu Hsuan) Lee, Author

ACKNOWLEDGEMENTS

I would like to acknowledge and thank all of those who made this possible and supported me. My advisor, Dr. Geoff Hollinger, who imparted wisdom, guidance, and provided me with an opportunity to work at the Robotic Decision Making Laboratory (RDML). Dr. Graeme Best, whose constant mentorship, valuable knowledge, and tenacity pushed me to be a better roboticist. I'd also like to thank the rest of RDML for the technical support and for being the best labmates anyone could ask for.

I would like to thank Bob Debortoli for convincing me to move out west to join the program and providing priceless advice. Special thanks to Ariel Kellogg for being a wonderful, lasting friend and organizing delicious family meals. Thank you Tucker, for being the cutest and best dog that you are. You're a goober bud. Also, thanks to Ameer Helmi, Nigel Swenson, and Ryan Quick for Dominion competition and friendship through trying times.

Lastly, I'm grateful to my father, mother, and my brother who have supported me time and time again.

This work was in part sponsored by DARPA under agreement #HR00111820044 and Office of Naval Research Grant N00014-17-1-2581. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF FIGURES (Continued)

LIST OF ALGORITHMS

Chapter 1: Introduction

Exploration of increasingly complex environments demands more flexible robotic capabilities. Developments in heterogeneous multi-robot systems have yielded carrier-passenger robot systems called *marsupial robots* (Fig. 1.1), in which highly-capable *carrier robots* transport and deploy one or more low-capability *passenger robots*. These marsupial robots can tailor their complementary capabilities to the challenges of exploring complex environments [3]. During exploration, these environments can contain multiple features of interest that the carrier robot may want to observe but are prohibitive or difficult to reach [4, 5]. In marine environments, a large ship may carry and deploy multiple heterogeneous robots to increase the rate of information gathering of features such as seafloor mines, adversarial vessels, and biological hotspots [6, 7, 8, 9]. In the case of urban environments [5, 10, 11], a team of ground carrier robots carrying aerial passenger robots may seek to explore features like ledges, vertical shafts, stairways, or other urban features that ground robots cannot easily reach [12]. The DARPA Subterranean Challenge [2] exemplifies real-world complex environments that marsupial robots are well suited for. Specifically, during the Urban circuit of the challenge, the stairs and vertical shafts in the SATSOP Nuclear Power Plant (Fig. 1.2) presented ideal locations for the carrier robot to deploy the passenger aerial drone.

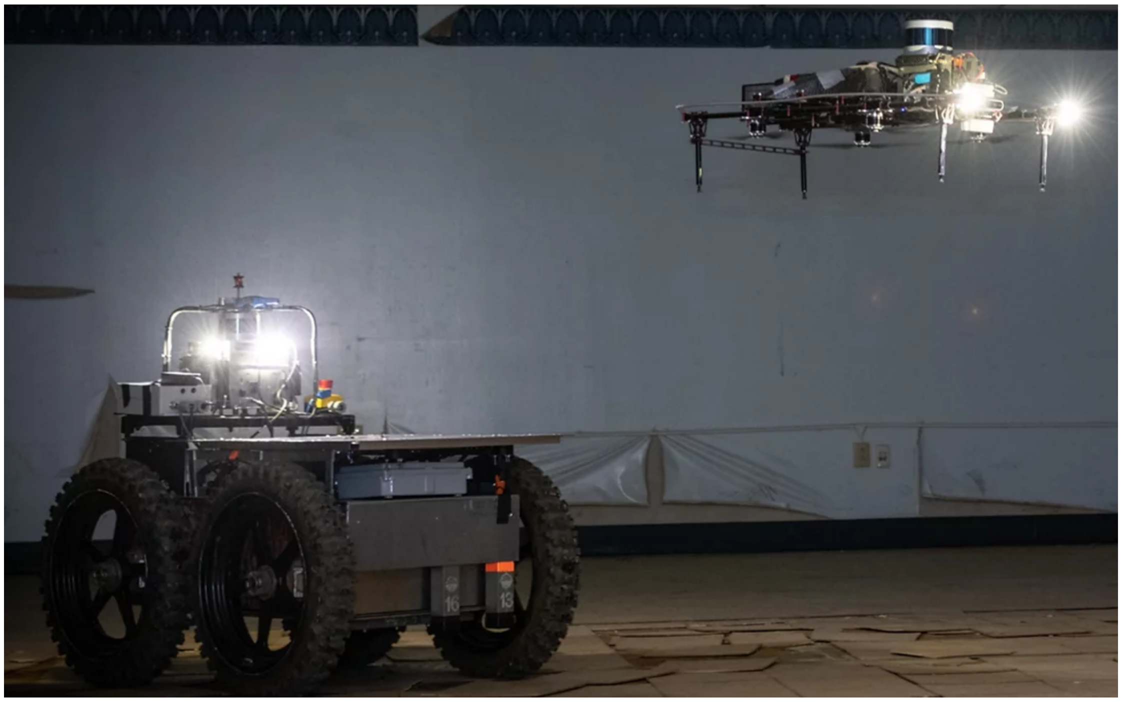


Figure 1.1: An example of a marsupial robot system by Team Explorer (CMU and OSU) [1]. The carrier robot on the left has increased payload capacity, better sensor capabilities, and longer battery life. The aerial passenger robot on the right is more lightweight and has better agility compared to the carrier robot.

As we scale to larger and more complex environments, it is necessary to expand these marsupial robot teams to comprise multiple carrier robots. This enables carrying larger teams of passenger robots and achieving a wider spatial coverage of passenger robot deployment locations. Critically, with the scaling of passenger robots and carrier robots, the number of deployment decisions can rapidly rise, placing an increasing amount of burden on the operator if in a teleoperation control mode. Additionally, with the restrictions in communication-denied environments, it becomes necessary to develop online passenger robot deployment methods for

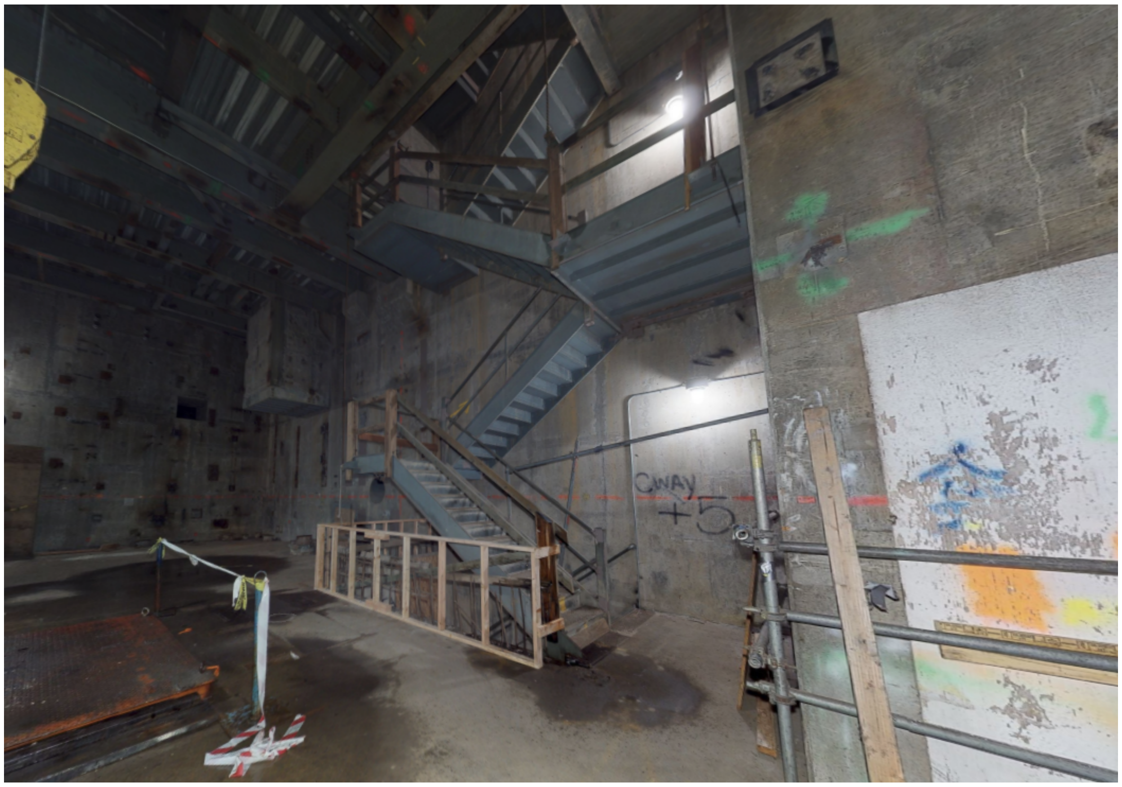


Figure 1.2: A potential deployment location for a passenger robot in the SATSOP Nuclear Power Plant [2]. A large carrier robot may encounter difficulties in ascending the staircase whereas an agile passenger robot drone would be able to traverse the obstacle easily.

these marsupial robot teams. Lastly, not only do the deployment methods need to account for varied numbers of carrier and passenger robots, they must scale to larger environments with higher potential deployment locations.

During these missions, the carrier robots would ideally deploy the passenger robots at locations that can maximize exploration coverage or information gain and make use of the passenger robot's complementary capabilities (e.g., flying in 3D). If a carrier robot deploys too early, it risks missing out on the potentially more valuable decision points later on. On the other hand, due to a late deployment, backtracking to a previous potential deployment location that turned out to be highly valuable is inefficient and undesirable.

To address the challenges above, we present passenger robot deployment algorithms for single and multiple marsupial robot teams that are able to reason over future expected rewards and conflicting passenger robot deployments in multiple marsupial robot teams. The goal of the work is to directly address the passenger robot deployment problem during exploration missions and enable online informed deployment decision-making for the carrier robots.

1.1 Thesis Contributions

In this thesis, we will present the following contributions to the field of multi-robot coordination with marsupial robots:

1. A single carrier robot deployment algorithm for multiple passenger robots, which builds on the Sequential Stochastic Assignment Problem [13], that maximizes the expected sum of the deployment rewards. The algorithm reasons over future expected rewards compared to the current location based on a known or estimated prior distribution.

2. A multiple carrier robot deployment algorithm for multiple passenger robots, which employs Monte Carlo Tree Search (MCTS) [14] to reason over passenger robot deployment conflicts and is coupled with a fast and effective problem-specific heuristic rollout policy based on the single carrier robot

deployment algorithm.

3. Drone deployment exploration simulated experiments with real-world data from the DARPA Subterranean (SubT) Challenge Urban circuit [1], where a ground robot seeks to deploy drones in exploration-valuable deployment locations.

4. Marine robot deployment simulations with Regional Ocean Modeling System (ROMS) temperature data, where marine robots are deployed in a large and expansive environment to collect biological information.

The contributions in this work provide novel methods for marsupial robot teams to make informed online decisions for passenger robot deployment using online long-horizon planners built on Monte Carlo tree search [14] and the solution to the Sequential Stochastic Assignment Problem [13].

1.2 Thesis Summary

The thesis is structured as follows.

Chapter 2 discusses prior and related work in optimal stopping, multi-robot coordination, marsupial robotics, and passenger robot deployment.

Chapter 3 formulates the deployment problem as a Sequential Stochastic Assignment Problem (SSAP) and presents the single carrier robot deployment algorithm.

Chapter 4 presents the multiple carrier robot deployment algorithm that deconflicts passenger robot deployments using MCTS.

Chapter 5 concludes the thesis and presents avenues for future work.

Chapter 2: Related Work

This chapter presents related work necessary for contextualizing this thesis and the key problem of general passenger robot deployments in marsupial robot systems. Sec. 2.1 introduces marsupial robots and the limitations of current passenger robot deployment algorithms. Sec. 2.2 presents background on optimal stopping theory, upon which our work relates to and draws inspiration from. Lastly, our work utilizes Monte-Carlo Tree Search (MCTS) to deconflict multiple passenger robot deployments. MCTS background is provided in Sec. 2.3.

2.1 Marsupial Robots and Passenger Robot Deployment

Marsupial robots are heterogeneous robot systems that consist of two or more different types of robots in carrier/passenger roles and typically consist of complementary but different capabilities [3]. These robot systems are termed "marsupial robots" based on their physical coupling, where the larger and more capable *carrier* robot transports a single or multiple *passenger* robots, which are lighter, smaller, and agile but less capable.

Traditional heterogeneous robot systems have loosely-coupled motion constraints that are not physically constricted and have been coordinated with distributed planners [15, 16]. In contrast, marsupial robots have tightly-coupled constraints that are physically imposed on the deployments of the passenger robots by the carrier robots. The increased complexity requires new planners that are able to handle tasks like coordination, deployment, retrieval, and manipulation. Some marsupial robot planners are able to organize and coordinate between multiple marsupial robotic systems to handle exploratory navigation actions [5, 17] but do not explicitly decide on optimal deployment locations for passenger robots. Other planners require known potential routes and deployment locations to decide optimal passenger robot deployment locations [18]. Another marsupial robot planner variant simplifies the deployment decision by deploying single passenger robots at fixed times such as the beginning of the mission [19].

The problem of planning deployment locations is key to the success of all of the above decision making, as a poor deployment location selection can result in passenger robots being unable to contribute towards the mission objectives. These deployment decisions have largely been left to human operators [20], who may not have adequate situational awareness to effectively make these decisions. One way to automate these decisions is to formulate a policy that triggers deployments based on predefined Boolean conditions, such as the discovery of a goal that is unreachable to the carrier ground robot [10, 19]. However, these relatively simple approaches are limited in applicability and do not reason over the long term value of decisions, particularly when there are multiple passenger robots to deploy. These limitations have been addressed by our work in this thesis.

Other related deployment algorithms include formulating the problem as a coverage-type problem [21, 22], as well as algorithms that combine more complex triggers such as power consumption and communication availability [23]. However, these approaches do not consider conflicts and dependencies between different deployment decisions. Finally, prior domain knowledge, such as marine flow fields [22], has been used to inform multiple passenger robot planners to maximize information gain. However, these types of historical information field data may not be readily available for other domains such as urban environments.

2.2 Optimal Stopping

Our deployment problem is closely related to optimal stopping theory [24], which considers problems that require deciding online the right time to carry out a particular action. A key example of an optimal stopping problem is the Classic Secretary problem, but it does not leverage prior information regarding future rewards [24]. However, the Cayley–Moser problem [25] reasons over predicted future rewards by considering the prior probability distribution. This has been applied to robotics problems by Lindhé and Johansson [26], who consider the problem of when to communicate by utilizing a multi-path fading communication model to make predictions about future stopping decisions. Our method is a generalization of the Cayley–Moser algorithm, adapted for the context of the deployment problem. Additionally, we formulate a policy that considers all future deployments, rather than just the single next deployment.

Optimal stopping variants have been generalized to multiple decision points. Das et al. [6] apply multi-choice optimal stopping theory for Autonomous Underwater Vehicles (AUVs) to collect multiple plankton samples that minimize the cumulative regret of the samples. They explore two applicable algorithms, the multi-choice hiring algorithm [27] and the submodular secretary algorithm [28]. Both of these algorithms seek to select the best observations and do not consider information such as the distribution of the observations. The sequential stochastic assignment problem (SSAP) [13] generalizes the Cayley–Moser algorithm and finds an optimal policy to maximize the expected sum of rewards from multiple assignments of agents to tasks. SSAP has been applied to other fields [29] but, to our knowledge, has not been utilized in robotics. A closely related body of work focuses on multi-robot task assignment [30]; however, these problems generally assume that task values are known a priori and are therefore not directly applicable to online problems, such as ours.

2.3 Monte Carlo Tree Search

MCTS [14] is rising in popularity as a high performing search algorithm for sequential decision making in robotics. This is in part due to its flexibility to optimize with respect to a general class of reward functions and its ability to naturally handle uncertainty [9, 31, 32, 33, 34, 35]. Best results are often achieved when replacing the default random rollout policy with problem-specific heuristics, which are directly leveraged to guide the best-first search procedure [36]. In robotics, a common problem-specific rollout policy is the greedy policy for information gathering [9, 32, 37]. For the deployment problem, the single-carrier SSAP solution discussed above [38] presents a useful heuristic for the multi-carrier generalization.

MCTS-based algorithms have also recently been proposed for the context of multi-robot systems. These include centralized algorithms where a single search tree represents the action space of all robots [35], and decentralized algorithms where each robot separately searches over a tree representing its own actions [32, 34]. Here, we focus on centralized contexts, and thus formulate a similar search tree to [35], but our proposed SSAP-based rollout policy is also applicable to the decentralized MCTS variants.

We leverage MCTS to reason over these conflicts and dependencies while borrowing ideas from the SSAP-based algorithm [38].

Chapter 3: Stochastic Sequential Assignment of Passenger Robots in a Marsupial Robot System

This chapter presents an informed deployment algorithm for a single marsupial robot system [38], which addresses the limitations of multi-robot planners discussed in the previous chapter. Furthermore, the single marsupial robot deployment algorithm for passenger robots sets the foundation for the multi-marsupial robot deployment algorithm that addresses the generalized case of passenger robot deployment discussed in the next chapter.

The complexities of heterogeneous marsupial robots enable a robot system to consist of flexible and complementary capabilities that are typically not afforded to single robot systems. However, the increase in capabilities and flexibility to overcome challenges come at a requirement for more in-depth planners which are able to reason over the deployment decisions of the passenger robots. As a marsupial robot system explores through the environment, ideally it would consider the current and historical information in order to inform the deployment decision of its passenger robots. Especially if the payload consists of multiple passenger robots, the deployment decision becomes non-trivial when trying to optimize for maximization of rewards from deployments.

We formulate the passenger robot deployment problem as a Sequential Stochastic Assignment Problem (SSAP) [13] (Sec. 3.1). We optimize the performance of passenger robot deployment to best explore an environment by proposing an algorithm that reasons over uncertainty by exploiting information about the prior probability distribution of features of interest in the environment (Sec. 3.2). The key feature of the algorithm is a recurrence relationship that defines a set of observation thresholds that are used to decide when to deploy passenger robots. We conducted simulated drone deployment exploration experiments on real-world data from the DARPA Subterranean challenge to test the SSAP algorithm. Our results show that our deployment algorithm outperforms other competing algorithms, such as the classic secretary approach and baseline partitioning methods, and is comparable to an offline oracle algorithm (Sec. 3.3). Finally, the implications of the result and the algorithm are summarized (Sec. 3.4).

3.1 Passenger Robot Deployment Problem Formulation

We consider a marsupial robot system traveling through an unknown environment that must make online decisions regarding when to deploy its passenger robots. At each possible deployment location, the robot must reason if the reward gained from deployment is expected to be more favorable than continuing onwards and deploying at a later location. We formulate the multi-robot deployment decision as a sequential stochastic assignment problem (SSAP). Multiple deployment decisions are made based on sequentially revealed random variables. The problem is formalized as follows.

3.1.1 Prior Distributions of Features in the Environment

We propose an environment that contains a set of point features, which may represent points of interest that are ideal for additional exploration by passenger robots, but are ill-suited for the carrier robot. As the carrier robot moves through the environment, it must make decisions to deploy or not deploy a passenger robot. There are assumed to be a total of N decision points, where N is known to the robot. The carrier robot houses R passenger robots of equal capability. At each decision point, the carrier robot may choose to deploy one passenger robot. Along the sequence of decision points, the independent observations of the number of features in the sensing area are denoted $(X_1, ..., X_N)$. Furthermore, the algorithm relies on knowing a prior distribution of the random variable X, denoted as $f(x)$. Later in Sec. 3.3, we present an example where the features of interest correspond to the number of exploration frontier cells observed by the ground robot with a prior distribution $f(x)$ defined as a Poisson distribution.

3.1.2 Deployment Decisions

At stage $j \in \{1, ..., N\}$, the carrier robot reaches a decision point, and the outcomes of all random variables $(X_1, ..., X_j)$, denoted $(x_1, ..., x_j)$, are known to the robot. Along the path, the robot must make an irreversible decision at each deploy location to deploy or continue on to deploy later. If the carrier robot decides to deploy, it assigns one passenger robot to the deployment location and returns a reward of x_j. If the carrier robot decides to continue, no reward is claimed at this stage. This process continues for the N stages. All passenger robots must be deployed by stage N, with the constraint of one deployment per stage.

3.1.3 Problem Statement

We define the set of stage indices, that indicates what stages the passenger robots were deployed at, as $(d_1, ..., d_R)$. The goal of the carrier robot is to maximize the expected sum of the reward returned from the deployment locations; i.e., find the optimal deployment sequence:

$$(d_1, ..., d_R)^* = \underset{(d_1,...,d_R)}{\arg\max}\, \mathbb{E}\Big[\sum_{r=1}^{R} x_{d_r}\Big]. \tag{3.1}$$

Here, x_{d_r} denotes the reward from passenger robot r and the expectation is taken over the distribution $f(x)$.

3.2 Online Passenger Deployment Algorithm

We present the online passenger deployment algorithm, which finds the optimal deployment policy in linear time that scales with the number of passenger robots and deployment locations. The algorithm is computed via dynamic programming, using a technique that stems from sequential stochastic assignment [13]. We consider the general case of R passenger robots to deploy. Finally, we provide an analysis of the optimality and runtime complexity of the algorithm.

3.2.1 Deployment of Multiple Passenger Robots

Our deployment algorithm precomputes a set of thresholds, which are a function of the distribution of the observations and the number of remaining stages, n. The current observed value x_j, defined in Sec. 3.1.1, at stage j is compared to the threshold values and informs the carrier robot whether or not to deploy now. At each stage j, we define a sequence of values $(p_1, ..., p_n)$, such that $p_i = 1$ for $n - r < i \leq n$, representing the remaining r passenger robots, and $p_i = 0$ for $1 \leq i \leq n - r$. This can be thought of as assigning a p_i at each stage to each x_j.

Specifically, the optimal policy for the passenger robot assignment is to assign p_i to the observed deployment value x_j, if x_j falls into an ith non-overlapping interval comprising the real line [13].

3.2.2 Deployment Thresholds

These non-overlapping intervals are separated by a set of thresholds, denoted as $a_{i,n}$. Each threshold $a_{i,n}$ is computed recursively and depends only on the prior distribution $f(x)$, as well as the number of remaining stages n. For stage j, where there are $n = N - j + 1$ stages remaining, there are a set of $n + 1$ thresholds, such that

$$-\infty = a_{0,n} \leq a_{1,n} \leq a_{2,n} \leq ... \leq a_{n,n} = \infty. \tag{3.2}$$

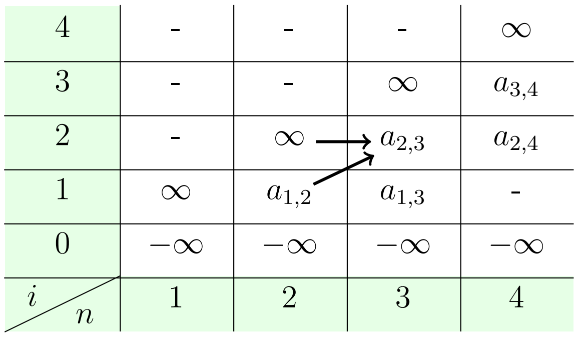
At stage j, the optimal choice is to assign p_i if the realization x_j of the random variable X_j is contained in $(a_{i-1,n}, a_{i,n}]$. For example, given $n = 4$ remaining stages and $r = 2$ robots to deploy, $p_3, p_4 = 1$ and $p_1, p_2 = 0$. If the observed value of x_j is $a_{2,4} < x_j \leq a_{3,4}$, then the decision is to deploy and returns a reward of $p_3 x_j = x_j$. At the next stage with $n = 3$ and $r = 1$, we only have $p_3 = 1$ and $p_1, p_2 = 0$. If the next observed value $x_j < a_{2,3}$, then there is no deploy action and returns a reward of $p_2 \cdot x_j = 0$.

An assignment of $p_i = 1$ to x_j results in a deployment of the passenger robot whereas an assignment of $p_i = 0$ to x_j leads to a non-deployment action. There is no need to differentiate between interval containment below the deploy threshold $a_{i-R,n}$ since all of the $n-R$ assignments of $p_i = 0$ result in the same non-deployment behavior. Thus, only R thresholds, instead of $n+1$ as shown in (3.2), need to be calculated for each stage n beyond $n = R$, and this reduces the complexity from quadratic [13] to linear in N, as discussed later in Sec. 3.2.3.

The threshold $a_{i,n+1}$ is defined as the expected value, if there are n stages remaining, of the quantity to which the ith smallest p is assigned [13]. This formulates the recurrence relationship

$$\begin{aligned} a_{i,n+1} &= \Pr(x_n < a_{i-1,n})a_{i-1,n} \\ &\quad + \Pr(a_{i-1,n} < x_n < a_{i,n}) \times \mathbb{E}(x_n | a_{i-1,n} < x_n < a_{i,n}) \\ &\quad + \Pr(x_n > a_{i,n})a_{i,n} \qquad (3.3) \\ &= a_{i-1,n} \int_{-\infty}^{a_{i-1,n}} f(x)dx + \int_{a_{i-1,n}}^{a_{i,n}} x f(x)dx + a_{i,n} \int_{a_{i,n}}^{\infty} f(x)dx, \qquad (3.4) \end{aligned}$$

where $-\infty \cdot 0 = 0$ and $\infty \cdot 0 = 0$. In both (3.3) and (3.4), the second term is for the case where x_n lies within the ith interval, and therefore p_i receives the value



4	-	-	-	∞
3	-	-	∞	$a_{3,4}$
2	-	$\infty \longrightarrow$	$a_{2,3}$	$a_{2,4}$
1	∞	$a_{1,2} \nearrow$	$a_{1,3}$	-
0	$-\infty$	$-\infty$	$-\infty$	$-\infty$
i \ n	1	2	3	4

Figure 3.1: The method for the recurrence threshold calculation and terms relationship in Eq. (3.3), for $R = 2$ and $N = 4$. $a_{i,n}$ is the threshold for the ith non-overlapping interval when there are n stages left to go. The term $a_{2,3}$ is calculated from the $a_{i-1,n}$ term, which is $a_{1,2}$, and the $a_{i,n}$ term, which is $a_{2,2} = \infty$. $a_{1,4}$ does not need to be calculated since $R = 2$.

of x_n. The first term is for the case where x_n is below the ith interval, meaning that x_n is assigned to a $p_k < p_i$, and thus p_i is assigned in a later stage with an expected value of $a_{i-1,n}$. Similarly, the third term is for the case where x_n is above the interval and p_i has an expected future assignment of $a_{i,n}$.

The recurrence process is illustrated in Fig. 3.1, where the cells in the table can be computed from left to right by reusing the previous values, as indicated by the arrows. Furthermore, an outline of the algorithm is provided in Alg. 1. The outer loop iterates through the N stages while the inner loop iterates through only R thresholds. The algorithm's recurrence process has a computational complexity of $O(NR)$. In the $R = 1$ single passenger robot deployment case, the threshold calculations are identical to the thresholds used in the Cayley–Moser optimal stopping problem [25].

Algorithm 1 *SSAP* optimal policy threshold calculation for passenger robot deployment

Input: Number of stages: N, Number of robots: R,
Prior distribution: $f(x)$
Output: SSAP Thresholds: $a_{i,j}$

```
1: for $j = 0, 1, 2, \dots, N$ do
2:     $a_{j,j} = \infty$
3:     for $i = j - R, \dots, j - 1$ do
4:         $a_{i,j} \leftarrow Eqn.\ (3.4)$
```

3.2.3 Analysis

The dynamic programming proceeds by iteratively solving optimal subproblems for $a_{i,n+1}$ using the recurrence relationship in (3.4), and thus computes the optimal set of thresholds $a_{i,n}, \forall i, n$. The full proof of the results that these subproblems are optimal, and that these thresholds lead to an optimal online assignment policy, is presented in [13]. The proof proceeds by induction, and relies on Hardy's Theorem [39], which states that the optimal assignment between two sets with a sum-product objective is to pair the smallest values in each set, then the next smallest, and so on until the largest values are paired.

As illustrated in Fig. 3.1 and Alg. 1, there are $O(NR)$ sub-problems to be computed, where N is the number of stages and R is the number of passenger robots to deploy. The integral (3.4) is computed once per sub-problem, thus the computation time is $O(NRF)$, where F is the time to compute (3.4).

3.3 Experiments and Results

In order to evaluate our proposed SSAP passenger robot deployment algorithm, we performed drone deployment exploration experiments using simulated data and real-world data from the DARPA Subterranean challenge, against various other deployment strategies. A single carrier robot was utilized for each deployment operation with varying numbers of passenger robots.

3.3.1 Comparisons of Alternative Deployment Strategies

Different deployment strategies were employed, each aiming to maximize the number of features captured during drone deployment, to test the efficacy of our SSAP deployment algorithm. These deployment strategies were adapted from optimal stopping algorithm variants to explicitly handle multiple robot deployments. As such, for these comparison methods, the carrier robot path is naively divided into R equal partitions in the case of multi-robot deployment and treat each partition independently from each other. Lastly, the *Random* deployment algorithm was selected as a baseline deployment strategy, as other naive deployment strategies (e.g., greedy deployment) returned similar results. The deployment algorithms are described below:

- *Sequential Assignment (SSAP)*: Performs our method as described in Sec. 3.2.1 and considers the entire path as a singular deployment, without partitions. The prior distributions are described below in Sec. 3.3.2.1 and Sec. 3.3.3.1.

- *Oracle*: Selects the top R deployment locations across all the partitions, with perfect knowledge of the world in advance.
- *Partition Oracle*: Selects the best deployment location within each partition, with perfect knowledge of the world in advance.
- *Partition Cayley–Moser*: Performs as described in Sec. 3.2.1 when $R = 1$, in each partition.
- *Partition Classic Secretary*: An optimal stopping variant that only observes for the first N/eR, where e is Euler's number, of decision points and then selects the next value that is higher than the max value observed in the observation phase [24]. The algorithm runs within each partition.
- *Random*: Selects a decision point randomly within each partition.

The experiments were conducted on a standard desktop computer, with an i5-4690K CPU and 16 GB of RAM.

3.3.2 Capturing Poisson-Distributed Features

3.3.2.1 Experimental Setup

A carrier robot travels through a 2D simulated world containing features of interest distributed as a stationary Poisson point process. At decision point j, the carrier robot detects features within a circular sensing area centered at the current location. The number of features x_j detected within a sensing area are modeled as a Poisson distribution with probability mass function

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!}, \text{for } x \in \{0, 1, ...\}. \tag{3.5}$$

Here, the Poisson distribution $f(x)$ in (3.5) is an example of a distribution used in the threshold calculations in (3.4). The rate of this Poisson distribution, λ, is the expected number of features of interest in the robot sensing area. The *SSAP* thresholds are calculated by substituting (3.5) into (3.4), yielding

$$a_{i,n+1} = a_{i-1,n} \sum_{x=0}^{\lfloor a_{i-1,n} \rfloor} \frac{\lambda^x e^{-\lambda}}{x!} + \sum_{x=\lceil a_{i-1,n} \rceil}^{\lfloor a_{i,n} \rfloor} x \frac{\lambda^x e^{-\lambda}}{x!} + a_{i,n} \left[1 - \sum_{x=0}^{\lfloor a_{i,n} \rfloor} \frac{\lambda^x e^{-\lambda}}{x!} \right]. \tag{3.6}$$

3.3.2.2 Results

An example of a trial of the algorithm selection process is illustrated in Fig. 3.2. The *Oracle* algorithm optimally selects the top $R = 3$ rewards. *SSAP* decided to deploy and select the reward in an optimal fashion in the first two decision cases. In comparison, the *Classic Secretary* algorithm does not reason over possible future observations and locally selects a decision point in each partition.

The results of *SSAP* through repeated trials are very encouraging. Trials with three passenger robots and $n = 60$ stages are shown in Fig. 3.3. During each trial, the algorithms' results are compared to the maximum possible number of features captured by the *Oracle*. On average, *SSAP* performed within 96% of the *Oracle*.

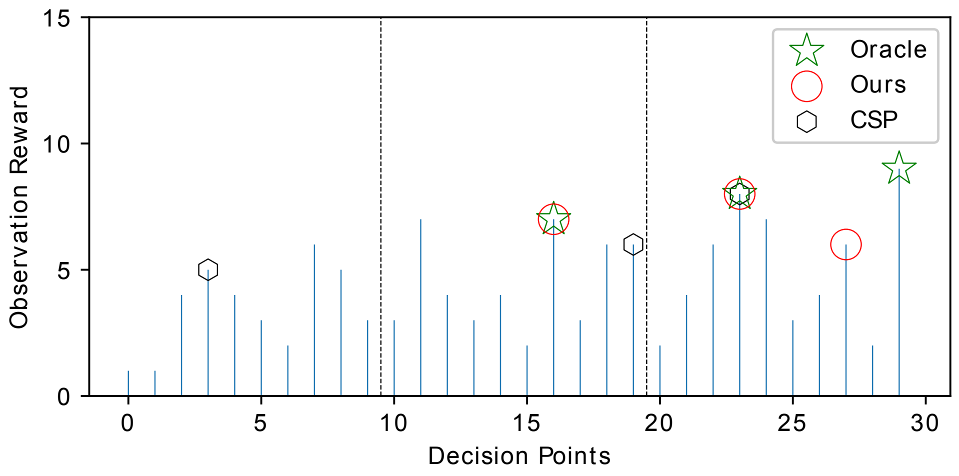


Figure 3.2: Illustrative example of the deployments for three passenger robots to be deployed over 30 decision points. Our method (reward: 21) performs similar to the *Oracle* (reward: 24) that has full knowledge of the observation sequence in advance. The *Classic Secretary* (reward: 19) algorithm collected less reward, in part due to being constrained by having one deployment in each of the three partitions (dotted lines).

Also, *Partition Oracle* is not perfect like the *Oracle* but performs similarly when compared to *SSAP*.

The partitioned *Cayley–Moser* algorithm has a lower performance than *SSAP* since the calculated thresholds are only local to each partition and partitioned algorithms are constrained to one deployment location per partition. *SSAP* is able to account for the expected future values over the total number of stages, whereas the partitioned *Cayley–Moser* locally calculated the thresholds only for the number of stages in a partition.

Lastly, we show the scalability of *SSAP* to R passenger robots in Fig. 3.4. The offline computation of the thresholds is linear in N stages and R number of robots. In practice, the computation of the thresholds takes on the order of seconds. As R

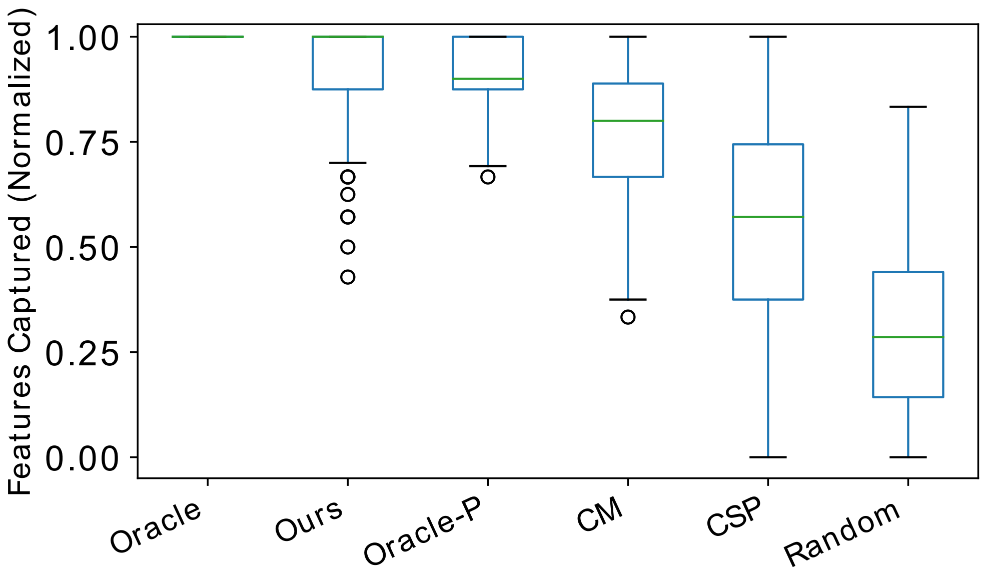


Figure 3.3: Algorithm comparisons with three passenger robots over 150 trials. *Partition Oracle* is denoted as Oracle-P. CM is *Cayley–Moser* and CSP is *Classic Secretary*. The simulated environment was populated with random features.

increases, the performance of *SSAP* clearly outperforms other algorithms. *SSAP* is mathematically equivalent to *Cayley Moser* in the case where $R = 1$ and yields identical results, as discussed in Sec. 3.2.1.

3.3.3 Aerial Robot Deployment for Exploration

3.3.3.1 Experimental Setup

Recorded LiDAR data from the Urban Circuit of the DARPA Subterranean Challenge [2] provided a real-world scenario to test the efficacy of our passenger deployment algorithm. Multiple drone-carrying ground robots autonomously explored

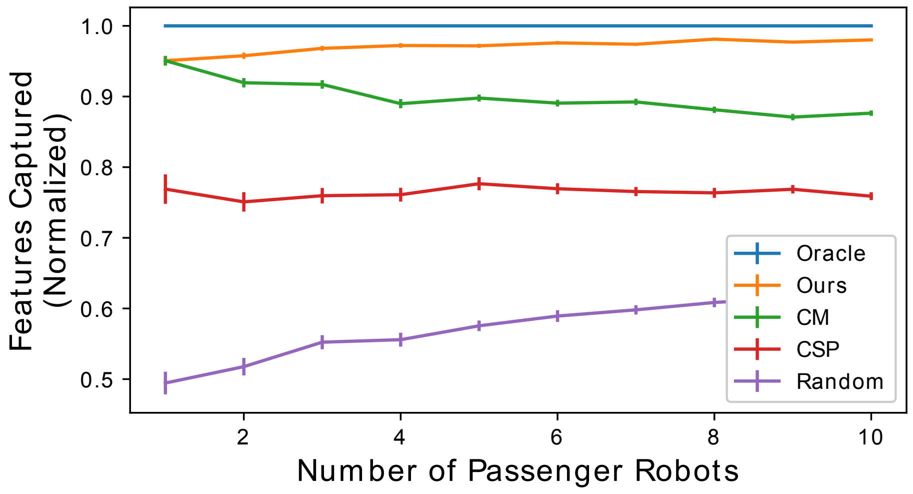


Figure 3.4: Algorithm comparisons with average utility, as a percentage of the *Oracle*, over R passenger robots and 150 trials. Standard error of the mean (SEM) error bars are displayed. The simulated environment was populated with random features.

the Satsop Nuclear Power Plant in Elma, Washington. For many teams in the competition, a drone deployment was manually triggered by the operator. Identifying an ideal deployment location required constant attention from the operator, already burdened with other tasks. Furthermore, it would be advantageous to utilize prior knowledge from the first robot entering the environment or information from a similar environment.

An occupancy grid of the world was generated from the collected LiDAR data using OpenVDB [40]. Using these data, a deployment decision location was set every 2.5 meters along the path that the robot travelled throughout the environment. We defined the value of a deployment location based on the total number of frontier cells within a 10 m radius of the ground robot's location. Frontier cells in

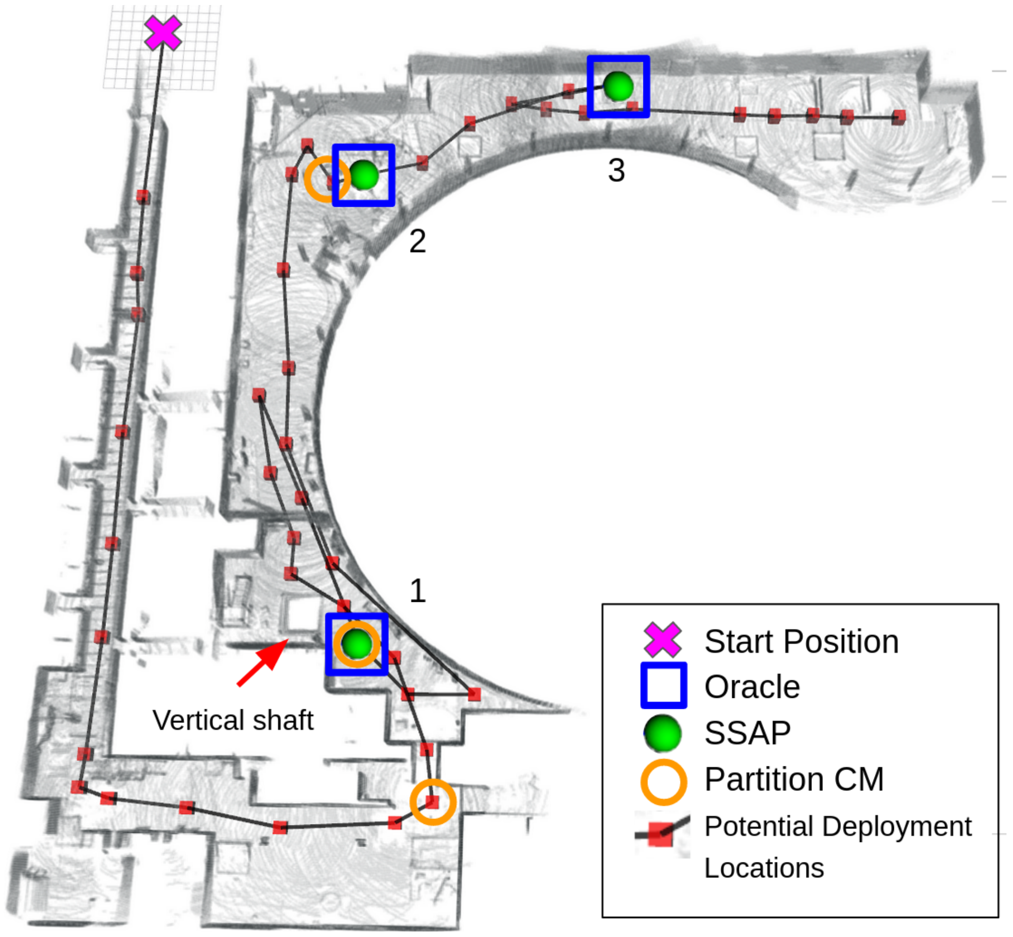


Figure 3.5: Deployment locations along a 105 m path in the Alpha environment from the DARPA SubT Urban Circuit. The *SSAP* algorithm decides to deploy in the optimal locations. Deploy location 1 is adjacent to a vertical shaft, reachable only by an aerial drone. Location 2 has a higher ceiling and is a larger space than the first *CM* location. Lastly, location 3 leads to the rest of the environment and provides high exploration value.

the occupancy grid are defined to be free cells neighboring at least one unknown cell [41]. Furthermore, we filter out frontier cells that are considered accessible to the ground robot. Only frontier cells that are above 1 m and below 0.1 m of the ground robot's position are included. These filtered frontier cells may represent openings such as shafts and ledges that are inaccessible to the ground robots but are ideal for aerial drones.

Different distributions, representing different prior knowledge, for $f(x)$ in (3.4) were defined in the calculation of the *SSAP* thresholds. In addition to the Poisson distribution, we compared two other distributions utilized by *SSAP*: (1) the histogram distribution and (2) Conway-Maxwell-Poisson (CMP) distribution. First, a distribution was generated from a histogram of the deployment values encountered during the robot's run and can be considered the ground truth distribution for that specific robot's path. Secondly, the CMP distribution generalizes the Poisson distribution and can better handle over/under-dispersion [42], with the probability mass function

$$f(x) = \frac{\lambda^x}{(x!)^\nu} \frac{1}{Z(\lambda, \nu)}, \text{for } x \in \{0, 1, ...\} \tag{3.7}$$

where $Z(\lambda, \nu)$ is a normalization constant. The parameters ν and λ were estimated using a Mean-Squared-Error fit to the histogram data. Similar to the Poisson distribution in Sec. 3.3.2.1, the CMP distribution in (3.7) can be substituted into (3.4) to generate the *SSAP* thresholds.

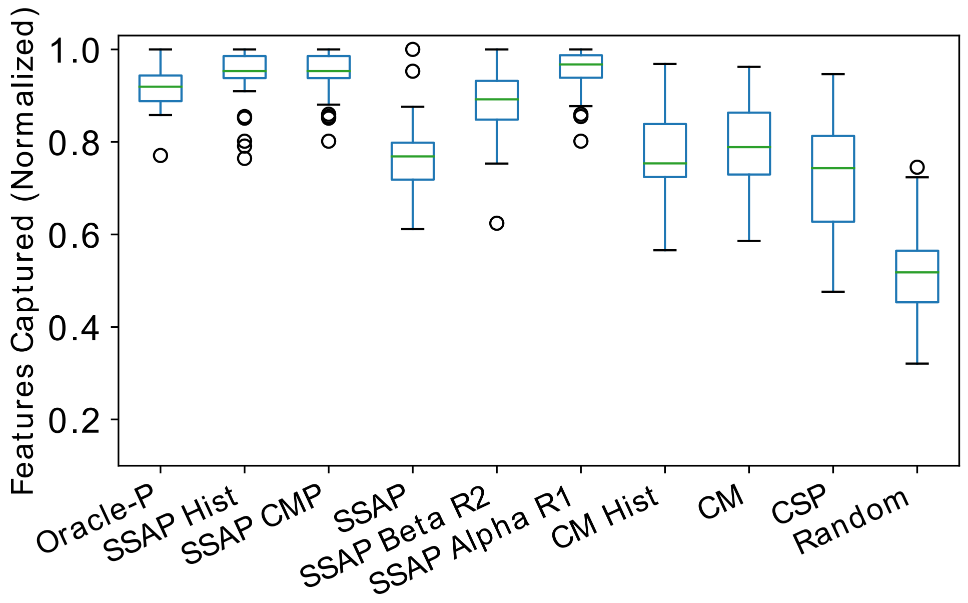


Figure 3.6: Algorithm comparison results from real data of a robot (R2) exploring Alpha environment. The *SSAP Beta R2* algorithm utilizes histogram data as the prior distribution from the same robot in another environment (Beta). The *SSAP Alpha R1* algorithm uses the histogram data from another robot (R1) in the same environment.

3.3.3.2 Results

The *SSAP* algorithm performed strongly in the real-world experiments and outperformed the competing deployment algorithms. The CMP distribution was able to better model the higher variability of real-world deployment values than the Poisson distribution. As seen in Fig. 3.6, *SSAP* using the CMP distribution performs within 87% of the oracle. While slightly less than the performance of the ground truth histogram distribution (91% of oracle), the *SSAP-CMP* distribution outperforms the *SSAP-Poisson* distribution by 18%. For the remaining experiment variations, *SSAP-CMP* is comparable to the *SSAP-Hist* algorithm and outperforms *SSAP-Poisson* by at least 10% and in one case, up to 28%.

We explored two cases of inter-robot information transfer to take advantage of prior domain knowledge: one where the histogram from one robot is used by another robot in the same environment and another where the data from another environment are used. The results of the same environment, inter-robot histogram transfer (*Alpha R1*) can be seen in Fig. 3.6. The *SSAP* algorithm using the histogram distribution from another robot performs comparably to the high-performing CMP and ground truth histogram distribution, above 90% of the oracle on average. In the other case, the histogram distribution from a different environment performed on average around 80% of oracle. Despite a lower performance than inter-robot histogram transfer, the inter-environment histogram transfer (*Beta R2*) can still provide better results than *SSAP-Poisson* or the other baseline methods, *partition Cayley–Moser* and *partition CSP*.

3.4 Summary

In this chapter, we provided a passenger deployment algorithm for a single carrier robot that can accommodate scaling numbers of passenger robots. The deployment algorithm generates a set of thresholds based on a known prior knowledge of the distributions of features of interest in the environment. Finally, these thresholds inform the carrier robot at each deployment location for the passenger robot deployment decision. Our results on simulated data of Poisson-distributed features and real data from the DARPA Subterranean challenge showcased the efficacy of our deployment algorithm compared to other deployment strategies. However, while this algorithm provides optimal performance for a single carrier robot with multiple passenger robots, the deployment algorithm is not able to account for deployment conflicts between multiple carrier robots. In the next chapter, we present work that leverages Monte Carlo tree search in order to address the multi-carrier robot deployment conflicts while utilizing the SSAP deployment algorithm from this chapter as an optimal rollout heuristic.

Chapter 4: Passenger Robot Deployments from Multiple Carrier Robots using Monte Carlo Tree Search

The previous chapter discussed the advantages and flexibility of a single marsupial robot team, as well as incorporating an intelligent passenger robot deployment decision-maker in the exploration planning. However, in a large and complex environment, a single marsupial robot system may not be sufficient or even capable of exploring and accomplishing the mission in a satisfactory amount of time. This can be accomplished by scaling the number of robots to meet the demands of larger and more complex environments. For a single marsupial robot team, the scaling can be managed by increasing the number of passenger robots transported by the carrier robot. However, this approach has a limit based on the carrier robot's payload constraint and deployment from a single carrier robot.

In this chapter, we explore scaling the number of marsupial robot teams, which both increases the number of carrier robots and the transported passenger robots. We present algorithm which provide the framework for a generalized deployment algorithm for any number of carrier and passenger robots. We specifically address the problem of planning the deployment times and locations of the carrier robots to best explore an environment while reasoning over uncertain future observations and rewards. Our work in single marsupial robot teams proposed an optimal, polynomial-time solution to *single*-carrier robot systems. However, the

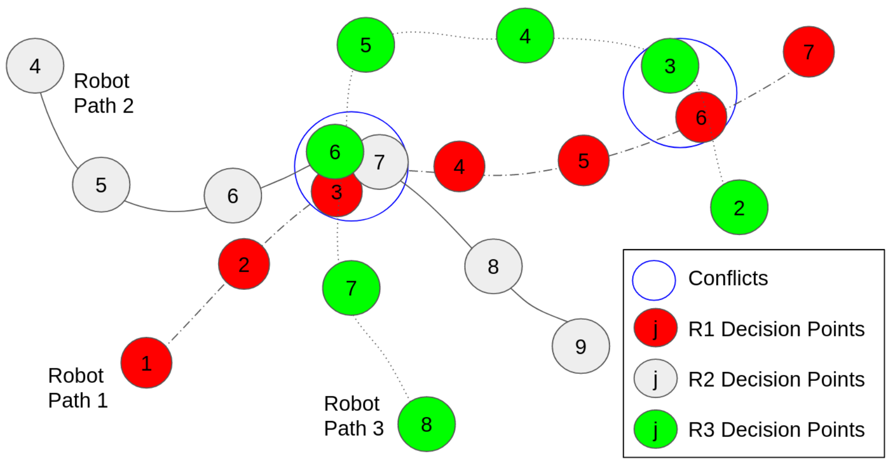


Figure 4.1: A deployment problem example with $R = 3$ carrier robots. Each colored circle represents a deployment decision location with the associated stage j. Here, overlap conflicts can be seen inside the blue circles. The first overlap conflict would be defined as $o_1 = \{[r_1, 3], [r_2, 7], [r_3, 6]\}$ and the second conflict defined as $o_2 = \{[r_1, 6], [r_3, 3]\}$.

multiple-carrier robot deployment problem is fundamentally harder as it requires addressing conflicts and dependencies between deployments of multiple passenger robots. We propose a centralized heuristic search algorithm for the multiple-carrier robot deployment problem that combines Monte Carlo Tree Search with a dynamic programming-based solution (Sec. 3.2) to the Sequential Stochastic Assignment Problem as a rollout action-selection policy. Our results with both procedurally-generated data and data drawn from the DARPA Subterranean Challenge Urban Circuit show the viability of our approach and substantial performance improvements over alternative algorithms.

4.1 The Multiple Marsupial Robot Deployment Problem

We consider a marsupial robot system consisting of multiple carrier robots, each carrying multiple passenger robots. A centralized decision maker must make online decisions regarding when to deploy the passenger robots. These decisions must be made while reasoning over both the reward gained from deploying immediately and the estimated rewards for deploying at an unknown later deployment location. The decision maker must also consider potential deployment conflicts by reasoning over the loss of reward caused by multiple deployments in close proximity. We formalize this problem as follows.

4.1.1 Marsupial Robot Team Decision Points

The multi-robot team consists of R carrier robots denoted $(r_1, r_2, ..., r_R)$, with each carrier robot carrying D passenger robots to deploy. Along each carrier robot's path, there are N stages that represent possible deployment locations. For each carrier robot r, there are a sequence of N random variables $(X_1^r, X_2^r, ..., X_N^r)$, which represent deployment rewards associated with each decision point (ignoring the conflicts introduced below in Sec. 4.1.2). We assume the random variables x are distributed according to a known prior distribution $f(x)$ and are assumed to be independent. At stage $j \in \{1, ..., N\}$, the carrier robot reaches a decision point, and the outcomes of all random variables $(X_1^r, X_2^r, ..., X_j^r)$, denoted $(x_1^r, x_2^r, ..., x_j^r)$, have been observed and are known to the decision maker.

For each carrier robot, a subset of the N stages may be invalid deployment locations; for example, due to staggered start times between the carrier robots. To maintain consistent indexing between the different carrier robots, we denote the realized observations at an invalid deployment stage j for a robot r as $x_j^r = \varnothing$.

4.1.2 Deployment Conflicts

As illustrated in Fig. 4.1, some possible deployment locations may be in close proximity to each other. However, in many applications, it is undesirable to deploy passenger robots in close proximity to each other since it may, for example, lead to inefficient exploration of the environment. We refer to groups of decision points in close proximity with each other as *conflicts*. Each conflict is defined by an overlap $o \in \mathcal{O}$, where $\mathcal{O}$ is the set of all overlaps. Each overlap $o \in \mathcal{O}$ is defined as a set of robot–stage pairs $o = \{[r_1, j_1], [r_2, j_2], ...\}$, which implies that the deployment location for carrier robot r_1 associated with stage j_1 is in conflict with the robot–stage pairs with o. For simplicity, we focus on addressing problems with overlap definitions where each carrier robot r does not have conflicts with itself at later stages (i.e., $r_i \neq r_j, \forall [r_i, j_i] \neq [r_j, j_j] \in o$), although the proposed algorithm allows for relaxing this constraint.

If carrier robot r deploys a passenger robot at stage j, then a reward of $v_d^r = x_j^r / p$ is received, where $1/p$ is a penalty factor associated with conflicts. Specifically, we define p as the number of selected passenger robot deployments that are within the same conflict set as $[r, j]$. This penalty factor defines a loss of reward associated with making multiple deployments in close proximity (conflict) with each other.

We note that while we focus on this conflict penalty definition throughout this chapter, our proposed algorithm would also be applicable to other definitions such as set cover formulations or any other quantifiable penalties arising from passenger robot deployments.

4.1.3 Deployment Actions

The action of carrier robot r at stage j is defined as a_j^r. At each deployment location, carrier robot r has two possible actions: deploy the passenger robot, or continue to the next deployment location. The joint actions of all of the R carrier robots at stage j is defined as $A_j = \{a_j^1, a_j^2, ..., a_j^R\}$. If the carrier robot has no remaining passenger robots to deploy, the carrier robot defaults to a no-deploy action. At stage j, the decision maker knows the previous deploy actions of all carrier robots.

4.1.4 Problem Statement

The set of stage indices where the passenger robots are deployed by carrier robot r are defined as $d^r = (d_1^r, d_2^r, ..., d_D^r)$. Furthermore, the set of all of the deployed stage indices across all carrier robots is defined as $\pi = \{d^1, d^2, ..., d^R\}$. As such, the goal of the decision maker is to maximize the expected sum of the rewards returned from the chosen deployment locations; i.e., find the optimal deploy sequences

$$\pi^* = \arg\max_{\pi} \mathbb{E}\Big[\sum_{r=1}^{R}\sum_{d=1}^{D} v_d^r\Big], \tag{4.1}$$

where R is the number of carrier robots, D is the number of passenger robots per carrier robot, and v_d^r is the reward for each deployment after accounting for conflicts (penalty factor $1/p$). The expectation in (4.1) is with respect to the unknown outcomes $(x_1^r, x_2^r, ..., x_j^r)$ of random variables $(X_1^r, X_2^r, ..., X_j^r)$.

The key computational challenges presented by this problem lie in the need to reason over these unknown rewards of future deployment decisions and also the need to account for deployment conflicts, which lead to dependencies in the reward structure.

4.2 MCTS with Sequential Stochastic Assignment Rollout Policy

We propose a centralized algorithm for the multiple marsupial robot deployment problem that combines MCTS [14] with heuristics derived from solutions to SSAP [13]. MCTS is a general-purpose sequential decision making algorithm that admits problem-specific heuristics. As discussed in the previous chapter, an SSAP-based algorithm has been proposed as an optimal solution to the *single* carrier robot deployment problem, here that SSAP-based algorithm serves as a strong heuristic to guide our MCTS-based algorithm for the *multiple* carrier robot deployment problem, which requires additionally addressing deployment conflicts.

This section begins by providing an overview of the tree data structure and overall algorithm, followed by a detailed description of the key algorithmic components and a brief analysis of runtime complexity and optimality.

4.2.1 Search Tree and Algorithm Overview

At stage j, our online algorithm incrementally expands and searches over a search tree $\mathcal{T}_j$, which represents the search space for the remaining deployment decisions. The root of $\mathcal{T}_j$ represents stage j, while subsequent layers of $\mathcal{T}_j$ are associated with future stages $\{j+1, ..., N\}$. The outgoing edges of a vertex enumerate all possible joint actions A_i available to the team at stage i.

Pseudocode for the algorithm is provided in Alg. 2. The search proceeds by employing standard MCTS [14] (described further in Sec. 4.2.2) with an adapted rollout phase that exploits information provided by the optimal solution to the single carrier deployment problem to guide the search tree expansion (described further in Sec. 4.2.3). The algorithm does this while reasoning over the various inputs introduced in Sec. 4.1, which we summarize as follows:

- the current observations x_j^r by each carrier robot of the associated random variables,
- the prior belief distribution $f(x)$ for all possible future deployment locations,
- the previous actions of the carrier robots $(A_1, ..., A_{j-1})$, which influences potential future deployment conflicts,

Algorithm 2 Overview of the online deployment algorithm for selecting the deployment actions A_j at stage j.

```
1: $\mathcal{T}_j \leftarrow$ initialize MCTS tree
2: ▷ Incrementally expand the search tree $\mathcal{T}_j$
3: for fixed number of iterations do
4:     $n \leftarrow$ SelectNode($\mathcal{T}_j$)    ▷ Select node to expand using UCT [43]
5:     $n^+ \leftarrow$ ExpandTree($n$)    ▷ Add new child node
6:     $(x^r_{j+1}, ..., x^r_N)_{\forall r} \leftarrow$ SampleFutureObservations($f$)    ▷ Sample from the prior belief distribution
7:     $(A_j, ..., A_N) \leftarrow$ SSAP_Rollout($n^+, (A_1, ..., A_{j-1}), (x^r_{j+1}, ..., x^r_N)_{\forall r}, \mathcal{O}$)    ▷ SSAP heuristic: See Sec. 4.2.3
8:     $v \leftarrow$ EvaluateSolution($(x^r_1, ..., x^r_N)_{\forall r}, (A_1, ..., A_N), \mathcal{O}$)    ▷ Evaluate the selected action sequence
9:     BackpropagateReward($n^+, v$)    ▷ Update statistics along the path back to the root node
10: ▷ Extract the solution
11: $n^* \leftarrow$ best node in $\mathcal{T}_j$
12: $A_j \leftarrow$ first joint action in sequence for $n^*$
13: return $A_j$
```

- the set of deployment conflict overlaps $\mathcal{O}$, and
- the number of stages remaining: $N - j$.

The search concludes after a predefined number of iterations and returns the estimated best joint deployment action A_j to execute at stage j.

4.2.2 MCTS for Multiple Carrier Robot Deployments

Our deployment algorithm employs the standard MCTS algorithm [14] to incrementally expand and search over the search tree $\mathcal{T}_j$. We summarize MCTS as follows, and defer our main innovation—the inclusion of a new SSAP-based rollout policy—to Sec. 4.2.3. MCTS consists of four phases: *selection*, *expansion*, *rollout*, and *backpropagation*.

In the *selection* phase (Alg. 2 line 4) a path is followed through $\mathcal{T}_j$ from the root node to a leaf node n. As per the standard Upper Confidence Bound 1 applied to trees (UCT) selection policy [43], this path is created by recursively selecting child nodes that maximize an upper confidence bound:

$$\bar{v}(n') + c\sqrt{\frac{\ln t(n)}{t(n')}}, \tag{4.2}$$

where $\bar{v}(n')$ is an estimate of the average reward associated with the node n' that is being considered, $t(n)$ is the number of samples that have passed through the parent node n, $t(n')$ is the number of samples that have passed through the considered node n', and c is the exploration parameter (typically set as $\sqrt{2}$ if the rewards are scaled between 0 and 1). The purpose of following this upper confidence bound policy is to naturally balance between exploring the tree (i.e., pick nodes that have been considered fewer times) and exploiting the knowledge gained so far (i.e., pick nodes with high expected rewards).

In the *expansion* phase (line 5), a new child node n^+ is added to $\mathcal{T}_j$ that extends from the selected node n. This node n^+ is chosen arbitrarily from any of the feasible joint actions that may be performed from the state represented by n. For our context of the deployment problem, a joint action is considered feasible if all carrier robots do not deploy more than the number of passenger robots they are carrying, deployments only occur at valid deployment stages (see Sec. 4.1.1), and at stage N all passenger robots have been deployed.

In the *rollout* phase (lines 6–8), a sequence of feasible actions is generated according to some given policy and then evaluated with respect to the reward function. In standard MCTS, the given policy is a random policy. However, there is also an opportunity here to employ problem-specific heuristics here to improve the reward estimates and speed up the search. We discuss our proposed rollout policy below in Sec. 4.2.3.

Every node n in the tree maintains a reward average $\bar{v}(n)$ and a count $t(n)$ of the number of rollouts that pass through this node. These two quantities are updated during the *backpropagation* phase (line 9). The rollout evaluation v is merged into the averages stored at all nodes on the path from the expanded node n^+ back to the root node, and the associated counts are incremented. These updated statistics guide the selection phase in the following iteration, and thus the tree expansions are directly influenced by the rollout evaluations.

At the end of the search (lines 10–13), the best node n^* is picked as the node with the highest expected reward. The first joint action in the sequence associated with n^* defines the deployment actions to be executed at this stage j. The algorithm is online such that the algorithm is run again at stage $j + 1$ while using any new gathered information, particularly the previous deployment decisions and new observations.

4.2.3 SSAP Action Selection Rollout Policy

As discussed above, problem-specific heuristics can be used within the *rollout* phase to improve the reward estimates, which in turn improves the MCTS node selection and therefore the overall search performance. Ideal rollout policies should be both fast to compute and provide good reward estimates. For our deployment problem, we draw inspiration from a solution to the single carrier robot problem presented in Sec. 3.2. While it is an optimal solution to the single carrier formulation, for the multi carrier problem this strong guarantee does not hold due to the need to address the conflicts, which breaks the independence assumption. However, we adapt it to provide reasonable reward estimates for our generalized problem that can be computed quickly within the online search algorithm.

Both the single carrier solution (Sec. 3.2) and our adapted heuristic for the multiple carrier case are based on the solution to the Sequential Stochastic Assignment Problem (SSAP) [13]. The aim of SSAP is to find the optimal subset (with given cardinality) of a set of random variables whose realizations are revealed sequentially. The decision to include or exclude a random variable from the subset must be made immediately when its realization is revealed.

SSAP maps closely to our deployment problem, as the value of a deployment location is only revealed as the carrier robot arrives at a location, and decisions to deploy or not deploy at a decision point are irreversible. As a reminder from the previous chapter and for convenience, we briefly formalize the optimal solution to SSAP as follows (further details may be found in Sec. 3.2) and discuss how we adapt it to be used as an MCTS rollout policy.

The optimal SSAP policy [13] consists of precomputing a set of thresholds $\{a_{i,n}\}$ for each stage. The realization x_j^r of the random variable X_j^r is compared to the relevant thresholds to determine whether to deploy now or wait until a later decision point. These thresholds are computed by considering the number of stages N and the prior belief distribution for the rewards $f(x)$. Specifically, for $n = N - j + 1$ stages remaining, the thresholds are computed via the recurrence relationship

$$a_{i,n+1} = a_{i-1,n} \int_{-\infty}^{a_{i-1,n}} f(x)dx + \int_{a_{i-1,n}}^{a_{i,n}} xf(x)dx + a_{i,n} \int_{a_{i,n}}^{\infty} f(x)dx,$$

where $-\infty \cdot 0 = 0$ and $\infty \cdot 0 = 0$. The second term represents an expectation of reward for the case where the reward is revealed to be within the two thresholds and therefore the decision maker receives this reward. The first integral represents the expected future reward if the reward lies in a lower threshold, and the third integral is for the case where it lies in a larger threshold. The set of all thresholds can be computed in quadratic time. If there are $n - i$ passenger robots left to deploy at stage $j = N - n + 1$, then the optimal policy is to deploy if x_j^r is greater than the ith interval, $a_{i,n}$, and do not deploy otherwise.

However, while the SSAP deployment policy is optimal for a single carrier robot, it does not consider multiple carrier robots and potential conflicts between the deploy locations, as defined in Sec. 4.1.2. The conflicts can be partly accounted for by adjusting the belief reward distributions $f(x)$ at decision points that conflict with *known* prior deployments by multiplying by the $1/p$ penalty factor. However, addressing unknown future conflicts would require enumerating all permutations of future deployments, which is infeasible to achieve within a fast MCTS rollout policy. Therefore, we instead rely on future MCTS iterations to correct for any error in the reward estimates by this SSAP heuristic.

The MCTS rollout policy (Alg. 2 lines 6–8) also requires evaluating the reward that will be achieved by the selected sequence of actions. While for specific belief distribution structures, it may be possible to efficiently compute the expectations in (4.1) for a specific deployment sequence, in general this will be impractical. Instead, we suggest approximating this expectation via Monte Carlo sampling of the random variable realizations from the prior distribution $f(x)$. The actions executed by a rollout can be selected by following the SSAP policy with respect to a sampled realization of the random variables $(X_{j+1}^r, ..., X_N^r), \forall r$. Within MCTS, these reward estimates are then used within the backpropagation phase (line 9) to guide the subsequent tree expansions.

4.2.4 Analysis

The fully expanded search tree contains 2^{RN} tree nodes, where R is the number of robots, and N is the number of stages. As MCTS is an iterative anytime algorithm, the computation time is directly proportional to the number of iterations, which in practice is typically selected by the user. The runtime complexity of each MCTS iteration of our algorithm is dominated by the SSAP rollout computation, which is achieved in quadratic time in the number of stages remaining (Sec. 3.2), and the reward evaluation, which has runtime proportional to the number of samples used in the Monte Carlo sampling.

MCTS also provides strong convergence rate guarantees for the convergence to the optimal action selection [43]. In practice, MCTS performance can be further enhanced with the use of problem-specific rollout policy heuristics [36]. While the proposed SSAP policy is optimal in the single-carrier case with no conflicts, this optimality guarantee does not extend to the generalized multiple-carrier case we consider here as the conflicts break the independence assumption of the reward distributions. Therefore, SSAP is instead best used as a strong heuristic within a general search algorithm that provides optimality guarantees, such as MCTS as we have proposed here.

We note that the search tree $\mathcal{T}_j$ that is incrementally expanded by our algorithm branches on actions but not observations. Thus, the convergence guarantee is toward the optimal open-loop action sequence rather than closed-loop policy. The advantage of seeking the open-loop action sequence is a lower branching factor, although it may come at the cost of optimality in some contexts. Alternative MCTS algorithms have been proposed for addressing the policy search problem [44], and our SSAP rollout policy could readily be used within these generalized MCTS algorithms.

4.3 Experiments and Results

We evaluate our multi-carrier, multi-passenger robot deployment algorithm in simulated experiments with Poisson-distributed simulated data and real-world data from the DARPA Subterranean Challenge [2]. The goal in these scenarios is for the passenger robots to maximally observe a set of features of interest by deploying passenger robots at high value locations. Our deployment algorithm is compared to other deployment strategies in order to demonstrate the benefits of accounting for the conflicts and employing a problem-specific MCTS rollout policy.

4.3.1 Deployment Comparison Methods

In the following experiments, we compare the following deployment strategies:

- *MCTS SSAP Rollout*: Our method as described in Sec. 4.2.1 with the rollout policy described in Sec. 4.2.3.
- *MCTS Random Rollout*: A variant of our method, however with the standard random policy [14] during the *rollout* phase.
- *Single Robot SSAP*: An optimal online deployment strategy for a single carrier robot with multiple passenger robots. This deployment strategy is applied to each carrier robot independently, without consideration for the conflicts between the deployment locations.
- *Random*: A baseline method that selects D decision points randomly from the path of each carrier robot.

4.3.2 Capturing Poisson Distributed Features of Interest

4.3.2.1 Experimental Setup

A system of multiple marsupial robots, composed of carrier robots transporting multiple passenger robots, travels through a 2D simulated world. As each carrier robot travels along the path, the robot detects features of interests at each decision point j. The features of interest in the environment are distributed as a stationary Poisson point process. Accordingly, the number of features x_j^r detected by carrier robot r at stage j is modeled as a Poisson distribution with probability mass function

$$f(x) = \frac{\lambda^x e^{-\lambda}}{x!}, \text{ for } x \in \{0, 1, ...\}. \tag{4.3}$$

If the carrier robot decides to deploy at the decision point, a reward of $v_j^r = x_j^r / p$ is received. The set of overlaps $\mathcal{O}$ were randomly generated and kept constant throughout the trials.

To ensure the rewards were approximately in the range $[0, 1]$, the rewards were normalized by the 90th percentile of the belief distribution $f(x)$. The exploration constant c was set as $0.05 \times \sqrt{2}$, which encouraged more exploitation of the problem-specific rollout function.

4.3.2.2 Results

Fig. 4.2 shows results for simulated trials with $R = 4$ carrier robots, each carrying $D = 3$ passenger robots. Both the *MCTS SSAP Rollout* and *MCTS Random Rollout* algorithm captured more features than the *Single Robot SSAP* algorithm, especially as the number of overlaps increased. The performance difference showcases the advantages of MCTS in accounting for overlap conflicts between the carrier robots and avoiding reward penalties stemming from deployment conflicts that the *Single Robot SSAP* is unable to do. Furthermore, by using an optimal deployment policy for a single robot, the SSAP rollout policy serves as a better heuristic than the random rollout policy, highlighted by the general better performance. All methods consistently outperformed *Random*.

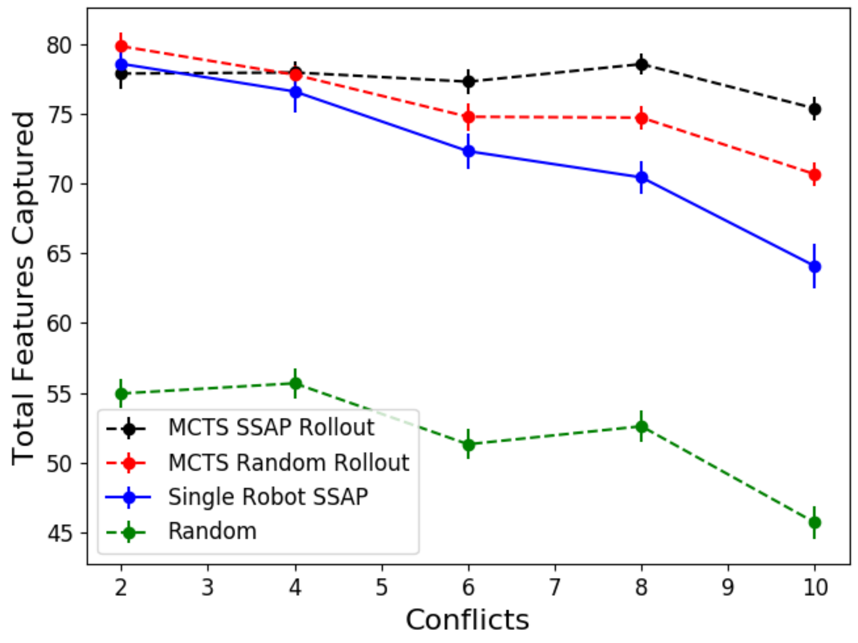


Figure 4.2: Algorithm comparisons with error bars (one standard error of the mean) of $R = 4$ carrier robots, each deploying $D = 3$ passenger robots, and traveling for $N = 10$ stages over 50 trials. The simulated environment was populated with random features. At each stage, MCTS ran for 10,000 iterations. The performance of *Single Robot SSAP* suffers as conflicts increases, due to its inability to account for overlaps. *MCTS SSAP Rollout* algorithm leverages the SSAP heuristic to improve over *MCTS Random Rollout* as the number of overlap conflicts increases, making more efficient use of the limited number of iterations.

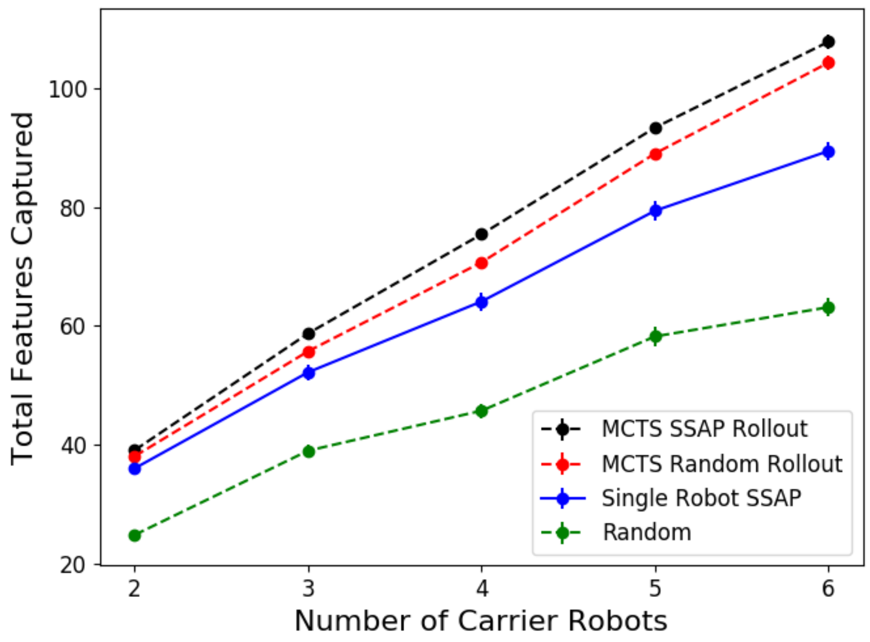


Figure 4.3: Algorithm comparisons with error bars (one standard error of the mean) of various numbers of carrier robots, each deploying $D = 3$ passenger robots. The simulated environment was populated with random features. The robots travel for $N = 10$ stages and encounter 10 potential overlap conflicts over 50 trials. At each stage, MCTS ran for 10,000 iterations.

We also show the scaling of our algorithm as the number of carrier robots R increases from $R = 2$ to $R = 6$. The number of MCTS iterations was kept constant across all trials. Despite the increasingly large joint action space, the MCTS algorithms consistently outperform the competing algorithms. By leveraging the search capabilities of MCTS, both *MCTS SSAP Rollout* and *MCTS Random Rollout* are able to deploy large amounts of passenger robots without heavy penalties incurred from conflicts. The potential for conflicts and overlap penalties increases with more robots in the environment, hindering the performance of *Single Robot SSAP*, which only considers each robot independently and does not account for overlap conflicts.

4.3.3 Subterranean Challenge Urban Circuit Data

To provide a real-world scenario for the application of the algorithm, we tested the efficacy of our deployment algorithm on recorded data from the Urban Circuit of the DARPA Subterranean Challenge [2]. A team of marsupial robots consisting of ground robots carrying aerial robots autonomously explore the Satsop Nuclear Power Plant in Elma, Washington. At the competition, several teams utilized marsupial robots, but the aerial robot deployment decisions were manually initiated by the human operator [45]. The operator needed to constantly look for valuable deployment locations for the passenger drones to deploy. As such, the operator was required to split their attention between other tasks and identifying ideal deployment locations. This placed an additional burden on the operator, on top of many other operational tasks required in the competition. Our algorithm aims to directly address the deployment problem autonomously, in order to relieve the burden from the operator.

4.3.3.1 Experimental Setup

Frontier cells in an occupancy grid were selected as a representation of features of interest (e.g., shafts, ledges, passageways) that are worthy for the passenger aerial robot to explore but cost-prohibitive to access by the ground robot. Using OpenVDB [40], an occupancy grid of the world was generated from the LiDAR data collected from the competition. The value of a deployment location is based on the total number of frontier cells captured within a 10 m radius of a ground robot's location. Cells are considered to be frontier cells if they are a free cell neighboring at least one unknown cell [41]. Additionally, we filter out easily accessible frontier cells by the ground robot and only retain frontier cells which are above 1 m and below 0.1 m the ground robot's position.

The carrier robots were deployed from the start location asynchronously and deployment decision locations were set every 100 seconds from the start of the hour-long run. The reward sequences for each robot were padded with $\varnothing$ observations for times where the carrier robot had not yet deployed from the start location, terminated the exploration early, or the robot's progress was stalled. For example, at stage $j = 3$, time-from-start $t = 300$, robot R1 has already started to explore the environment but robot R2 is still at the start gate, awaiting to be deployed into the environment. The observations of robot R1 at stage $j = 3$ would be $\{x_1^1, x_2^1, x_3^1\}$, whereas for robot R2, it would be $\{\varnothing, \varnothing, \varnothing\}$. For stages where robot r has an $\varnothing$ observation value, the deployment algorithm removes the child nodes containing deployment actions by robot r in the *expansion* phase and modifies the SSAP threshold calculations in the *rollout* phase to only consider N' stages remaining, where N' is the number of total non-$\varnothing$ stages. For the overlap conflicts between all three robots, any deployment points within 30 m (3 times the 10 m observation radius) of each other were considered to be conflicting and added to the overlap set $\mathcal{O}$.

4.3.3.2 Results

Our deployment algorithm was able to handle the potential deployment conflicts between the $R = 3$ robots and captured more features than the competing deployment algorithms. The *MCTS SSAP Rollout* algorithm captured 157 features, compared to *Single Robot SSAP*: 122 features, *MCTS Random Rollout*: 114 features, and *Random*: 68 features. The selected deployment locations for each robot are illustrated in Fig. 4.4. The set of deployment locations achieve good spatial coverage across the environment while targeting the areas with high aerial robot exploration value. As this was a real-world mission, multi-robot coordination planning was employed between the carrier robots, which naturally seeks to prevent path overlap between the robots to maximize exploration efficiency. Compared to the simulated data in Sec. 4.3.2, there is expected to be less non-uniformly dis-

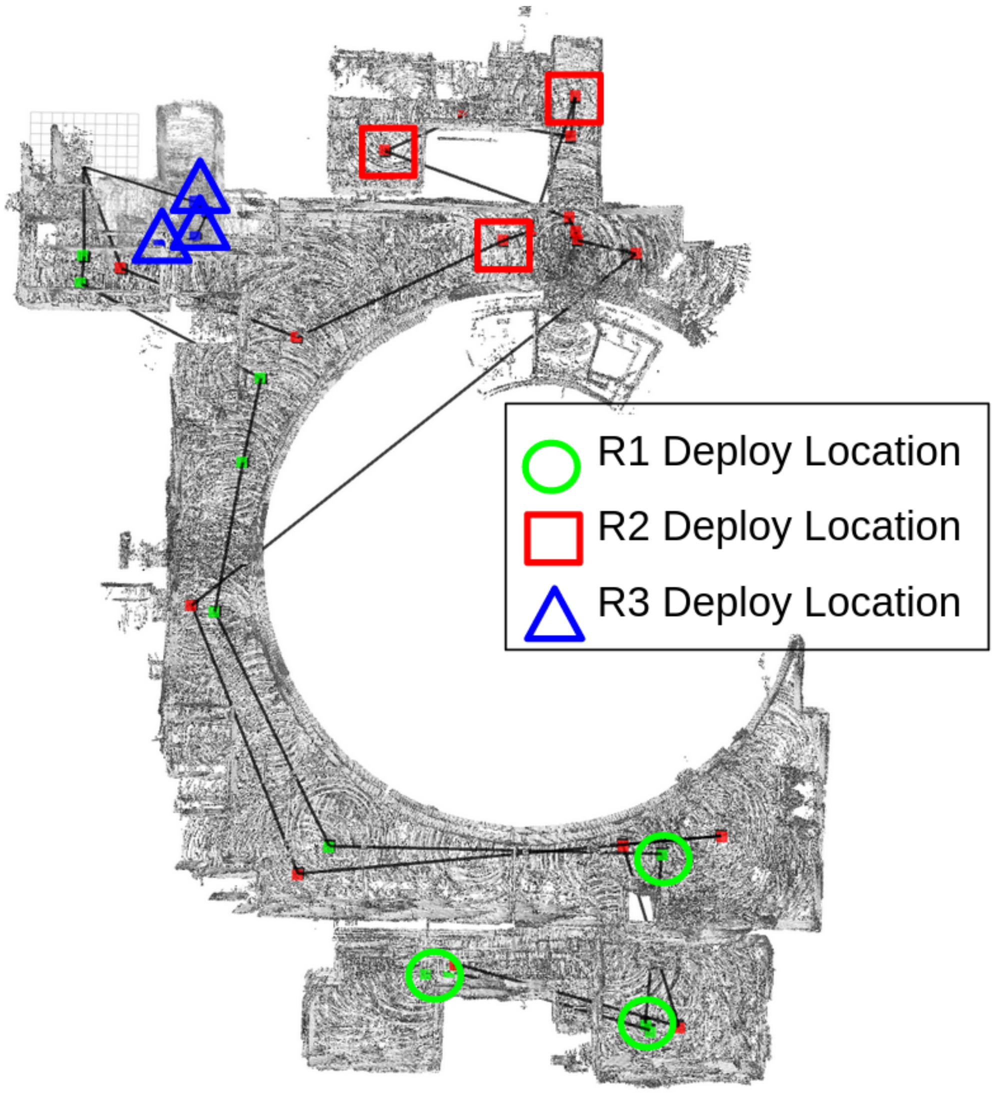


Figure 4.4: Deployment locations for each carrier robot in the Alpha environment from the DARPA Subterranean Urban Circuit selected by *MCTS SSAP Rollout* with 10,000 iterations and $N = 19$ stages. Robots 1 and 2 deploy in non-conflicting locations, avoiding the overlap conflicts. Robot 3 explores down a stairwell in the top left corner of the map and selects to deploy in those locations. The *Single Robot SSAP* algorithm decided to deploy in high value but conflicting locations.

tributed overlaps. However, there were several valuable locations that *Single Robot SSAP* selects to deploy at but without consideration for overlap conflicts, leading to penalties.

4.3.4 Marine Bioinformation Data collection

As shown in Sec. 4.3.3, the multi-marsupial robot deployment algorithm can effectively deploy multiple passenger robots in a complex urban environment. However, this environment has artificial boundaries in form of human-created structures, which effectively limit the size and scale of the environment. To showcase the efficacy of the algorithm in a larger environment with longer and varying path lengths, we conducted simulated deployments of marine robots on temperature data extracted from the Regional Ocean Modeling System (ROMS) in Monterey Bay, California. Ocean surface carrier robots traverse a pre-defined path throughout the bay, deploying underwater vehicles. Our algorithm not only scales with number of carrier and passenger robots, but is able to accommodate varying numbers of deployment locations. Furthermore, the deployment algorithm is structured as a framework that is able to utilize varying forms of prior knowledge as well as employ domain-specific heuristics. In this application, a Gaussian process was generated over the temperature data as the prior knowledge estimation, providing a distribution for the threshold calculation of the next deployment location to be visited.

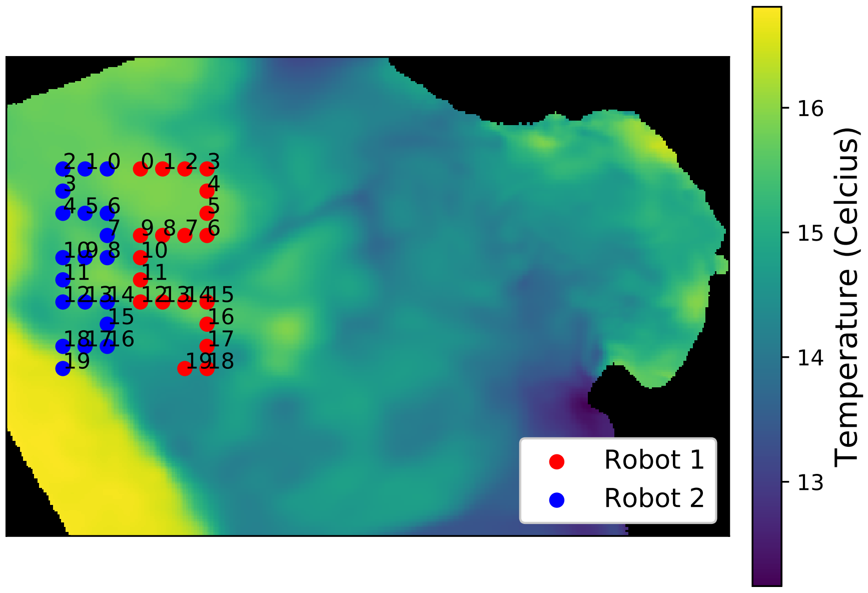


Figure 4.5: Regional Ocean Modeling System (ROMS) temperature data of Monterey Bay, CA. Lawnmower paths of two carrier robots with length $N = 20$ and different raster lengths are shown.

4.3.4.1 Experimental Setup

In order to simulate a marine deployment operation to collect biological information from upwelling fronts, Regional Ocean Modeling System (ROMS) temperature data of Monterey Bay, seen in Fig. 4.5, was utilized. Regions where cold, nutrient-rich water from the ocean floor meets warm surface waters present biologically informative areas [46] and are ideal locations for passenger robot deployment. Higher temperature locations where the carrier robot deploys present higher rewards than lower temperature locations, where the reward is defined as $V = \frac{\sum_{r=1}^{R} x_r}{R}$. Here, x_r

represents the temperature at the deployed location of the passenger robot r and R denotes the total number of passenger robots. Two carrier robot paths were simulated on lawnmower paths with varying raster lengths to create different stages where each robot would have potential overlaps with the other. The temperature data was modeled using a standard Gaussian process approach with a radial basis function (RBF) kernel and seeded with 10 randomly sampled locations from the data. The same comparison algorithms from Sec. 4.3.3 were used.

4.3.4.2 Results

Our results show that the deployment algorithm is able to accommodate operations with varying path lengths and utilize alternate forms of prior knowledge, such as a Gaussian process. Furthermore, this second case study shows that our algorithm can be generalized and applied to different domains, like this marine environment. Additionally, the algorithm can be reconfigured to utilize domain-specific reward functions such as temperature (as opposed to frontier cells for exploration).

As seen in Fig. 4.6, our deployment algorithm deployed the marine robots in locations which captured higher average temperatures than the competing algorithms of random deployments and the single marsupial robot deployment algorithm applied independently to each robot. As the number of deployment locations increases, the number of overlaps also increases, explaining the decrease in performance of the competing algorithms. The MCTS deployment algorithms are able to account for the increased conflicts in deployment locations of the passenger robots.

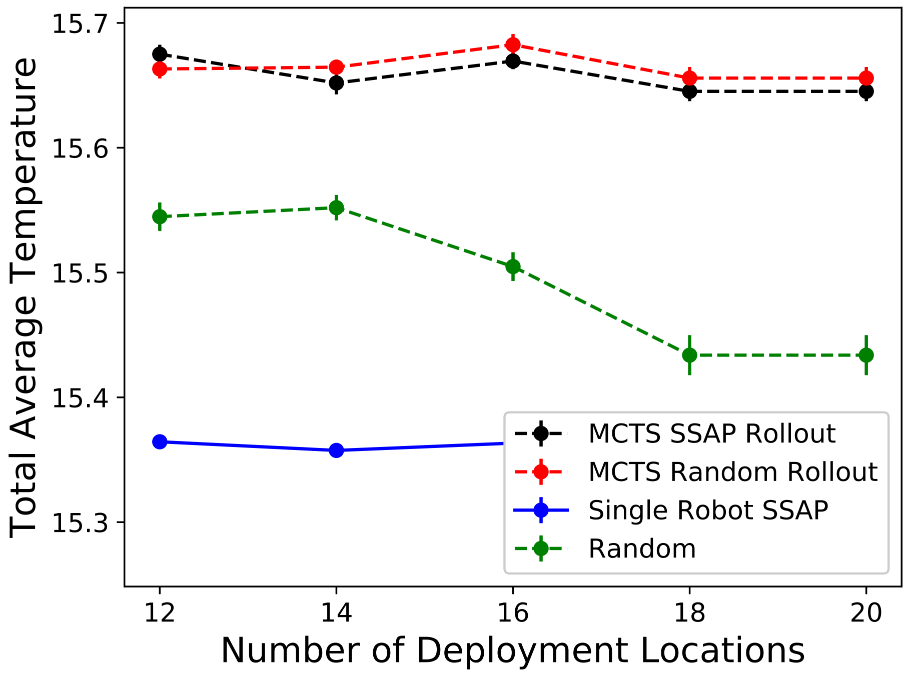


Figure 4.6: Temperature performance of the deployment algorithm with error bars (one standard error of the mean) of $R = 2$ carrier robots with up to $N = 20$ deployment locations. Over 50 trials, MCTS ran for 10,000 iterations at each stage. As the path length increased, the number of overlap locations also increased.

The *Single Robot SSAP* algorithm performs worse than the *Random* algorithm due to the deterministic thresholds likely selecting deployment locations with penalties but is unable to account for the penalty inducing conflicts.

The performances of the *MCTS SSAP rollout* and *MCTS Random rollout* algorithms are comparable across the number of deployment locations and can be attributed to a few reasons. Firstly, the SSAP rollout heuristic is sensitive to estimations of domain-specific prior belief. In this case, the standard Gaussian process parameters can be adjusted in the future to better represent the domain's underlying distribution, which may improve the performance of the SSAP rollout heuristic. Secondly, the effects of the SSAP rollout heuristic may be less effective due low data variance, which leads to narrow intervals from the thresholds. Lastly, the performance of the SSAP rollout heuristic highlights the deployment algorithm's sensitivity towards different assumptions of the prior belief distribution.

4.4 Summary

In this chapter, we presented a multi-carrier robot deployment algorithm based on MCTS which searches through the joint action space of multiple carrier robots to inform the team of deployments and avoid passenger robot deployment conflicts. The default MCTS random rollout policy is replaced with the optimal solution to the single carrier robot deployment case, discussed in the previous chapter. Our algorithm provides a framework for the generalized deployment problem, incorporating various forms of prior knowledge distributions of the environment's features of interest and is able to scale to multiple carrier and passenger robot systems. We showcased the performance and the varied application of the algorithm on Poisson-distributed simulated data, real data from the DARPA Subterranean challenge, and real marine ROMS temperature data from Monterey Bay.

Chapter 5: Conclusion and Future Work

The benefits and flexibility that marsupial robots provide for exploration of complex environments require capable planners which can account for the deployment of the passenger robots from the carrier robots. In this thesis, we have provided an initial problem formulation of the passenger robot deployment problem for marsupial robots and demonstrated the feasibility of the sequential assignment optimal policy to passenger robot deployments. Real-world operation of robot exploration tasks using marsupial robots can benefit from the application of our deployment algorithms to autonomously and intelligently decide deployment locations.

5.1 Summary of Contributions

In Chapter 3, we presented a novel formulation of the passenger deployment problem from a single carrier robot that is addressed with the solution to the sequential stochastic assignment problem [13]. Our simulation results showed that the deployment algorithm is able to reason over the future expected rewards of each deployment location and provide optimal performance.

In Chapter 4, we generalized the passenger deployment problem to multiple carrier robots with multiple passenger robots for a multi-marsupial robot system. In order to address the conflicting shortcomings of the single robot deployment algorithm, we utilized Monte Carlo tree search to explore the joint action space of the carrier robots for passenger robot deployment deconfliction. Our results demonstrated that our algorithms enable a significantly improved selection of robot deployment locations for exploration tasks, in comparison to a variety of alternative methods.

5.2 Future Directions

In the future, various avenues of improvements can be explored for the deployment algorithm. Since the current deployment algorithm considers only the carrier robot's actions and path, it would be important to incorporate the planned actions of the deployed passenger robots in order to inform the main deployment planner of potential future conflicts stemming from a potential deployment location and better model system rewards across both the carrier and passenger robots. Furthermore, the algorithm is scalable to varying numbers of deployment locations but requires the total number of locations to be fixed beforehand for the threshold generations. Future work can explore online replanning methods in order to extend the planner to an infinite horizon variant. Furthermore, a natural extension to the centralized approach would be to develop decentralized algorithms, such as by employing the SSAP rollout policy within Dec-MCTS [32], to enable deployment planning in communication sparse environments. It would also be interesting to plan over the space of closed-loop policies, rather than open-loop action sequence, which may be achieved with an MCTS generalization such as POMCP [44], as noted in Sec. 4.2.4. Also, different forms of conflicts and dependencies may exist between the deployment location rewards, which would lead to alternative reward function definitions, such as a set cover formulation.

The sequential stochastic assignment component of the deployment algorithm can also benefit from exploration of various extensions and variations of the problem. For example, it would be interesting to improve current distribution estimation [47], study cases where the prior feature distribution [48] or number of stages are not known, and consider a dependent relationship between consecutive observations.

Bibliography

[1] CMU/OSU, "SubT Team Explorer," 2021. Accessed on 03.22.2021.

[2] B. Allen and T. Chung, "Unearthing the subterranean environment," 2021. Accessed on 03.22.2021.

[3] R. R. Murphy, M. Ausmus, M. Bugajska, T. Ellis, T. Johnson, N. Kelley, J. Kiefer, and L. Pollock, "Marsupial-like mobile robot societies," *Proc. Int. Conf. on Autonomous Agents*, pp. 364–365, 1999.

[4] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic UAV and UGV system for precision agriculture," *IEEE Trans. on Robotics*, vol. 32, no. 6, pp. 1498–1511, 2016.

[5] J. Moore, K. C. Wolfe, M. S. Johannes, K. D. Katyal, M. P. Para, R. J. Murphy, J. Hatch, C. J. Taylor, R. J. Bamberger, and E. Tunstel, "Nested marsupial robotic system for search and sampling in increasingly constrained environments," in *Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC)*, pp. 2279–2286, 2016.

[6] J. Das, F. Py, J. B. Harvey, J. P. Ryan, A. Gellene, R. Graham, D. A. Caron, K. Rajan, and G. S. Sukhatme, "Data-driven robotic sampling for marine ecosystem monitoring," *Int. J. Robotics Research*, vol. 34, no. 12, pp. 1435–1452, 2015.

[7] F. Marques, A. Lourenço, R. Mendonça, E. Pinto, P. Rodrigues, P. Santana, and J. Barata, "A critical survey on marsupial robotic teams for environmental monitoring of water bodies," in *Proc. IEEE OCEANS*, 2015.

[8] M. Kalaitzakis, B. Cain, N. Vitzilaios, I. Rekleitis, and J. Moulton, "A marsupial robotic system for surveying and inspection of freshwater ecosystems," *J. Field Robotics*, vol. 38, no. 1, pp. 121–138, 2021.

[9] S. McCammon, G. Marcon dos Santos, M. Frantz, T. P. Welch, G. Best, R. K. Shearman, J. D. Nash, J. A. Barth, J. A. Adams, and G. A. Hollinger, "Ocean front detection and tracking using a team of heterogeneous marine vehicles," *J. Field Robotics*, vol. early access, 2021. DOI: 10.1002/rob.22014.

[10] P. G. Stankiewicz, S. Jenkins, G. E. Mullins, K. C. Wolfe, M. S. Johannes, and J. L. Moore, "A motion planning approach for marsupial robotic systems," in *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2018.

[11] J. Zhao, G. Liu, Y. Liu, and Y. Zhu, "Research on the application of a marsupial robot for coal mine rescue," in *Proc. Int. Conf. on Intelligent Robotics and Applications*, pp. 1127–1136, 2008.

[12] T. Rouček, M. Pecka, P. Čížek, T. Petříček, J. Bayer, V. Šalanskỳ, D. Heřt, M. Petrlík, T. Báča, V. Spurnỳ, *et al.*, "DARPA subterranean challenge: Multi-robotic exploration of underground environments," in *Proc. Int. Conf. on Modelling and Simulation for Autonomous Systems*, pp. 274–290, Springer, 2019.

[13] C. Derman, G. J. Lieberman, and S. M. Ross, "A sequential stochastic assignment problem," *Management Science*, vol. 18, no. 7, pp. 349–355, 1972.

[14] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," *IEEE Trans. on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.

[15] G. Best, W. Martens, and R. Fitch, "Path planning with spatiotemporal optimal stopping for stochastic mission monitoring," *IEEE Trans. on Robotics*, vol. 33, no. 3, pp. 629–646, 2017.

[16] M. Schwager, D. Rus, and J.-J. Slotine, "Unifying geometric, probabilistic, and potential field approaches to multi-robot deployment," *Int. J. Robotics Research*, vol. 30, no. 3, pp. 371–383, 2011.

[17] K. M. Wurm, C. Dornhege, B. Nebel, W. Burgard, and C. Stachniss, "Coordinating heterogeneous teams of robots using temporal symbolic planning," *Autonomous Robots*, vol. 34, no. 4, pp. 277–294, 2013.

[18] N. Mathew, S. L. Smith, and S. L. Waslander, "Planning paths for package delivery in heterogeneous multirobot teams," *IEEE Trans. on Automation Science and Engineering*, vol. 12, no. 4, pp. 1298–1308, 2015.

[19] J. C. Las Fargeas, P. T. Kabamba, and A. R. Girard, "Path planning for information acquisition and evasion using marsupial vehicles," in *Proc. IEEE American Control Conf. (ACC)*, pp. 3734–3739, 2015.

[20] M. J. Schuster, M. G. Müller, S. G. Brunner, H. Lehner, P. Lehner, R. Sakagami, A. Dömel, L. Meyer, B. Vodermayer, R. Giubilato, *et al.*, "The arches space-analogue demonstration mission: towards heterogeneous teams of autonomous robots for collaborative scientific sampling in planetary exploration," *IEEE Robotics and Automation Lett.*, vol. 5, no. 4, pp. 5315–5322, 2020.

[21] Y. Mei, Y.-H. Lu, Y. C. Hu, and C. G. Lee, "Deployment of mobile robots with energy and timing constraints," *IEEE Trans. on Robotics*, vol. 22, no. 3, pp. 507–522, 2006.

[22] J. Hansen, S. Manjanna, A. Q. Li, I. Rekleitis, and G. Dudek, "Autonomous marine sampling enhanced by strategically deployed drifters in marine flow fields," in *Proc. MTS/IEEE OCEANS*, 2018.

[23] H. Lee and B. Lee, "Path planning based on probabilistic roadmap for initial deployment of marsupial robot team," *Institute of Control, Robotics and Systems*, vol. 24, no. 1, pp. 80–89, 2018.

[24] T. S. Ferguson, *Optimal Stopping and Applications*, ch. Finite Horizon Problems, p. 2.1–2.15. UCLA, 2008.

[25] L. Moser, "On a problem of Cayley," *Scripta Math*, vol. 22, pp. 289–292, 1956.

[26] M. Lindhé and K. H. Johansson, "Exploiting multipath fading with a mobile robot," *Int. J. Robotics Research*, vol. 32, no. 12, pp. 1363–1380, 2013.

[27] Y. Girdhar and G. Dudek, "Optimal online data sampling or how to hire the best secretaries," in *Proc. IEEE Canadian Conf. on Computer and Robot Vision*, pp. 292–298, 2009.

[28] M. Bateni, M. Hajiaghayi, and M. Zadimoghaddam, "Submodular secretary problem and extensions," in *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pp. 39–52, Springer, 2010.

[29] V. Shestak, E. K. Chong, A. A. Maciejewski, and H. J. Siegel, "Probabilistic resource allocation in heterogeneous distributed systems with random failures," *J. of Parallel and Distributed Computing*, vol. 72, no. 10, pp. 1186–1194, 2012.

[30] G. A. Korsah, A. Stentz, and M. B. Dias, “A comprehensive taxonomy for multi-robot task allocation,” *Int. J. Robotics Research*, vol. 32, no. 12, pp. 1495–1512, 2013.

[31] W. Chen and L. Liu, “Pareto Monte Carlo tree search for multi-objective informative planning.,” in *Proc. Robotics: Science and Systems*, 2019.

[32] G. Best, O. Cliff, T. Patten, R. R. Mettu, and R. Fitch, “Dec-MCTS: Decentralized planning for multi-robot active perception,” *Int. J. Robotics Research*, vol. 38, no. 2-3, pp. 316–337, 2019.

[33] D. Bauer, T. Patten, and M. Vincze, “Monte Carlo tree search on directed acyclic graphs for object pose verification,” in *Proc. Int. Conf. on Computer Vision Systems*, pp. 386–396, 2019.

[34] M. Corah and N. Michael, “Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice,” *Autonomous Robots*, vol. 43, no. 2, pp. 485–501, 2019.

[35] B. Kartal, J. Godoy, I. Karamouzas, and S. J. Guy, “Stochastic tree search with useful cycles for patrolling problems,” in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 1289–1294, 2015.

[36] S. James, G. Konidaris, and B. Rosman, “An analysis of Monte Carlo tree search,” in *Proc. AAAI Conference on Artificial Intelligence*, 2017.

[37] M. Lauri, N. Atanasov, G. Pappas, and R. Ritala, “Active object recognition via Monte Carlo tree search,” in *Proc. IEEE ICRA Workshop on Beyond Geometric Constraints*, 2015.

[38] C. Y. H. Lee, G. Best, and G. A. Hollinger, “Optimal sequential stochastic deployment of multiple passenger robots,” in *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2021.

[39] G. Hardy, J. Littlewood, and G. Polya, *Inequalities.* Cambridge University Press, 1934.

[40] K. Museth, J. Lait, J. Johanson, J. Budsberg, R. Henderson, M. Alden, P. Cucka, D. Hill, and A. Pearce, “OpenVDB: An open-source data structure and toolkit for high-resolution volumes,” in *Proc. Int. Conf. on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 2013.

[41] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. IEEE Int. Symp. on Computational Intelligence in Robotics and Automation (CIRA)*, pp. 146–151, 1997.

[42] G. Shmueli, T. P. Minka, J. B. Kadane, S. Borle, and P. Boatwright, "A useful distribution for fitting discrete data: revival of the conway–maxwell–poisson distribution," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 54, no. 1, pp. 127–142, 2005.

[43] L. Kocsis, C. Szepesvári, and J. Willemson, "Improved Monte-Carlo search," tech. rep., University of Tartu, 2006.

[44] D. Silver and J. Veness, "Monte-Carlo planning in large POMDPs," in *Advances in Neural Information Processing Systems 23* (J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, eds.), pp. 2164–2172, Curran Associates, Inc., 2010.

[45] S. Scherer *et al.*, "Resilient and modular subterranean exploration with a team of roving and flying robots," *[To appear] Field Robotics*, 2021.

[46] S. McCammon and G. A. Hollinger, "Topological hotspot identification for informative path planning with a marine robot," in *2018 IEEE Int. Conf. on Robotics and Automation (ICRA)*, pp. 4865–4872, 2018.

[47] A. J. Lee and S. H. Jacobson, "Sequential stochastic assignment under uncertainty: estimation and convergence," *Statistical Inference for Stochastic Processes*, vol. 14, no. 1, pp. 21–46, 2011.

[48] S. C. Albright, "A Bayesian approach to a generalized house selling problem," *Management Science*, vol. 24, no. 4, pp. 432–440, 1977.