Probabilistic Scheduling for Multi-Robot Systems and Automation Applications

by

Giovanni D'urso

A thesis submitted in fulfilment of the
requirements for the degree of Doctor of Philosophy

Supervisor: Prof. Robert Fitch

at the
School of Mechanical and Mechatronic Engineering
Faculty of Engineering and Information Technology
University of Technology, Sydney

November 2023

Certificate of Original Authorship

I, Giovanni D'urso declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the Faculty of Engineering and Information Technology at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signature: Production Note:
Signature removed prior
to publication.

Date: 10/11/2023

Probabilistic Scheduling for Multi-Robot Systems and Automation Applications

by

Giovanni D'urso

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

Robots are anticipated to be useful in diverse environments and applications. This trend shown by the substantial investment, media coverage, and research in fields such as defence, home robots, space operations, logistics, and agriculture. Despite ongoing advancements, we still have yet to see the widespread use of robotics in practical settings. Robotics excel at addressing specific, well-defined problems that can lead to considerable economic gains, such as working at ports, driving on mapped roads, and working in warehouses with infrastructure installed. Yet, the broader integration of robotics into the remaining practical large-scale problems is hampered by their inherent complexities. These often involve teams of multiple robots, have long durations, have large spatial scales, include uncertainty, consist of long sequences of tasks, can involve humans, and have ill-defined objectives. This additional complexity requires a more considered approach to producing practical solutions that can be applied to solve these applications.

Existing approaches to multi-robot planning under uncertainty are limited primarily due to their inadequate handling of the unpredictable nature of these problems, which impacts both their practicality and scalability. Conventional methods, such as exact constraint-based or naive sampling-based methods, often inflate the perceived complexity of these problems, rendering them seemingly intractable. On the other hand, more sophisticated statistical methods, such as stochastic optimisation or partially observable Markov decision process methods, aim to overcome these limitations; however, they often require computation that renders them feasible for only the smallest problems. Such methods can be applied through extensive offline computation, but this is unsuitable as problems increase in complexity and are impractical for real-time field operations. The root of these issues lies in a lack of fundamental understanding of how to tackle different types of uncertainty. This thesis hypothesizes that a more granular understanding of uncertainties is crucial to developing methods that are both effective in practice and capable of scaling to the demands of complex real-world robotic applications.

In response, this thesis introduces effective strategies for real-world multi-robot and automation planning challenges by conceptualising them as probabilistic scheduling problems. This perspective allows a deeper understanding of the effects of uncertainty and the impact of problem scale, growth and the interaction between variables. Our main idea is that approaching these problems with this lens allows the development of principled planning algorithms to address the challenges of the practical scheduling of multi-robot and automation applications. This is due to identifying where specifically the uncertainty occurs in the problem and which dimensions/components are the most significant. With this understanding, it becomes possible to use predictive methods and decompositions to produce practical and feasible solutions.

To present this approach, we model and produce solutions for four concrete, real-world scenarios. Each problem demonstrates uncertainty within different aspects of standard scheduling challenges. The first is scientific information gathering with a heterogeneous team of agents exploring a Europa surface analogue considering objective uncertainty caused by a noisy map prior. The second is planning a sequence of operations for a team of autonomous diving floats and surface vessels to jointly map the seabed floor, with ocean currents causing uncertain constraints. The third is scheduling teams of logistics workers (agents) in an order-picking warehouse servicing dynamic, uncertain real-time orders throughout the day. The fourth is producing week-long schedules for using green hydrogen produced by uncertain wind forecasts, which requires considering uncertain resource allocations.

This exploration across this broad collection of examples leads to the beginning of a new theory of robotics scheduling that delineates four fundamental components of stochasticity: objectives, constraints, tasks, and resources. We introduce novel techniques to manage these stochastic elements that draw from our practical applications. We show through algorithmic analysis, simulation, and hardware trials that our methods that account for the number of agents and stochastic elements of the problem outperform methods that do not include these considerations. By solving these specific problems, we can open up new fields of application and research that promise considerable positive economic and social impacts. The analytical framework and solutions we propose will serve as a valuable resource for developing pragmatic multi-robot planning and autonomous scheduling solutions in the future.

Acknowledgements

Engineering problems are under-defined. There are many solutions, good, bad and indifferent. The art is to arrive at a good solution. This is a creative activity involving imagination, intuition and deliberate choice.
Sir Ove Arup
This thesis was completed with the assistance and support of many friends, family and colleagues. First, I would like to thank my supervisor, Prof. Robert Fitch. We have worked together for many years and through multiple degrees. I am thankful for your guidance both in technical academic endeavours and for improving my ability to communicate complex ideas in a convincing manner. Secondly, thank you to my co-supervisor Dr Chanyeol Yoo, for your advice and support with paper writing.

I have been lucky to work with some great people; thank you to everyone from the UTS and CDMRG for your inspiration and interesting weekly discussions. I am glad I met and worked with such capable people as James Lee, Fred Sukkar, Brian Lee, Cadmus To, Andre Nunez, Jennifer Wakulicz, Clyde Webster, Graeme Best and Tim Patten. A large portion of what I learned during this degree was from participating in the CDMRG reading group and other general discussions. I have had the luck and the opportunity to work with many great people while completing this degree. I would like to thank: the team from Premontion, Will Ried and Michael Paton from JPL, Michael Fueting, Basti, Carla, and Matti from DLR, Oscar Pizarro and Jackson Shields from Usyd, and my collaborators from Waterloo Armin Sadeghi and Stephen Smith. This thesis is made much stronger by a collection of ideas created through working with and alongside everyone.

Lastly, I would like to thank my family and friends. I know I have been absent and distracted for several years due to being deep into work; I appreciate the care and support you have given me throughout this process. I'm sure there are only so many times you can hear me talk about warehouses, lamenting about ocean currents, trying to make sense of the wind, and making lines on graphs make sense. Thank you to Amy, Adam, Anthony, Christine, Holmes, Steve, and Sharpie. Lastly, thank you to my family for caring for me despite being a pain at times. I truly appreciate it.

Contents

List of Figures

List of Tables

List of Algorithms

List of Problems and Remarks

Nomenclature

Acronyms & Abbreviations

ACO Ant Colony Optimisation

AGV Autonomous Ground Vehicle

ASV Autonomous Surface Vessels

AUV Autonomous Underwater Vehicles

CHP Combined Heat and Power

CVRP Capacitated Vehicle Routing Problem

CVRPTW Capacitated Vehicle Routing Problem with Time Windows

DECOP Discoverable Edge Cost Orienteering Problem

Dec-MCTS Decentralised Monte Carlo Tree Search

Dec-POMDP Decentralised Partially Observable Markov Decision Process

DOF Degrees Of Freedom

D-UCT Discounted-UCT

DVRP Dynamic Vehicle Routing Problem

FIFO First In First Out

GP Gaussian Process

IGMOP Intuition Guided Multi-robot Orienteering Problem

MCTS Monte Carlo Tree Search

MDP Markov Decision Process

MRTA Multi-Robot Task Allocation

MVMF Multi-Vessel Multi-Float

POMDP Partially Observable Markov Decision Process

PPV Positive Predictive Value

PRM Probabilistic RoadMap

POI Point Of Interest

TAMP Task And Motion Planning

TD-OP Time-Dependant Orienteering Problem

TOP Team Orienteering Problem

TSP Travelling Salesman Problem

UAV Unmanned Aerial Vehicle

UCB Upper Confidence Bound

UCT Upper Confidence bounds applied to Trees

UTS University of Technology Sydney

VPP Virtual Power Plant

VRP Vehicle Routing Problem

List of Publications

Conference Papers

1. **G. D’urso**, J. J. H. Lee, O. Pizarro, C. Yoo, R. Fitch, ’Hierarchical MCTS for scalable multi-vessel multi-float systems’, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, May, 2021.

2. **G. D’urso**, A. S, C. Yoo, S. Smith, R. Fitch ‘Distributed multi-agent equal partition algorithm for allocation in warehouse picking scenarios’, *Proceedings of IEEE International Conference on Automation Science and Engineering (CASE)*, August 2023.

Workshop Paper

1. **G. D’urso**, J. J. H. Lee, K. M. B. Lee, J. Shields, B. Leighton, O. Pizarro, ‘Field trial on ocean estimation for multi-vessel multi-float-based active perception,’ *IEEE International Conference on Robotics and Automation (ICRA)*, 1st Advanced Marine Robotics TC Workshop: Active Perception”, May 2021.

Journal Paper

1. **G. D’urso**, M. Füting, R. Fitch ‘Wind powered green hydrogen H2 production consumption scheduling’, *Journal of Energy Research*. ***to be submitted**

Awards

University

1. NASA Jet Propulsion Laboratory (JPL), Visiting Student Research Program, 2019
2. UTS School of Mechanical and Mechatronic Engineering 2019 Research Showcase - Second Prize (Poster)
3. UTS School of Mechanical and Mechatronic Engineering 2020 Research Showcase - First Prize (Oral Presentation)
4. Space Research Network (SRN) Higher Degree Research intern with German Aerospace Center (DLR), 2022

Chapter 1

Introduction

Robotics is a rapidly advancing field of research, and its impacts can be seen in a broad cross-section of modern society[1]. Emerging robotic systems have a wide variety of applications, with some of the most impactful applications being seen in the fields of environmental sustainability, agriculture, health, education, manufacturing, logistics, and autonomous transportation. Environmental sustainability can be improved through more sophisticated monitoring systems [2], which will allow us to understand and protect our environment and preserve our sites of cultural significance [3, 4]. Robotic agriculture systems such as those by [5, 6] can increase sustainable food production and reduce pesticide usage through spraying, weeding and planting automation. Another important field where robotics has an impact is health. Many robotic health applications range from robotic seals caring for an aging population [7] to allowing remote communities access to medical expertise [8]. Robotics can even benefit education; [9] taught high school students indigenous knowledge using low-cost robots, and [10] investigated the impact of using robots to engage socio-economically disadvantaged youths. Finally, some of the most visible impacts on our daily lives will stem from further advancements in manufacturing, logistics and autonomous transportation. In manufacturing, improvements can be seen in areas such as assembly line automation [11], quality control [12], and optimised dynamic production [13]. In logistics, innovations such as warehouse automation [14], last-mile delivery [15], and delivery drones [16] will make a notable difference. Lastly, autonomous transportation, including self-driving cars [17] and trucks[18], will reshape how industry and society operates [19].

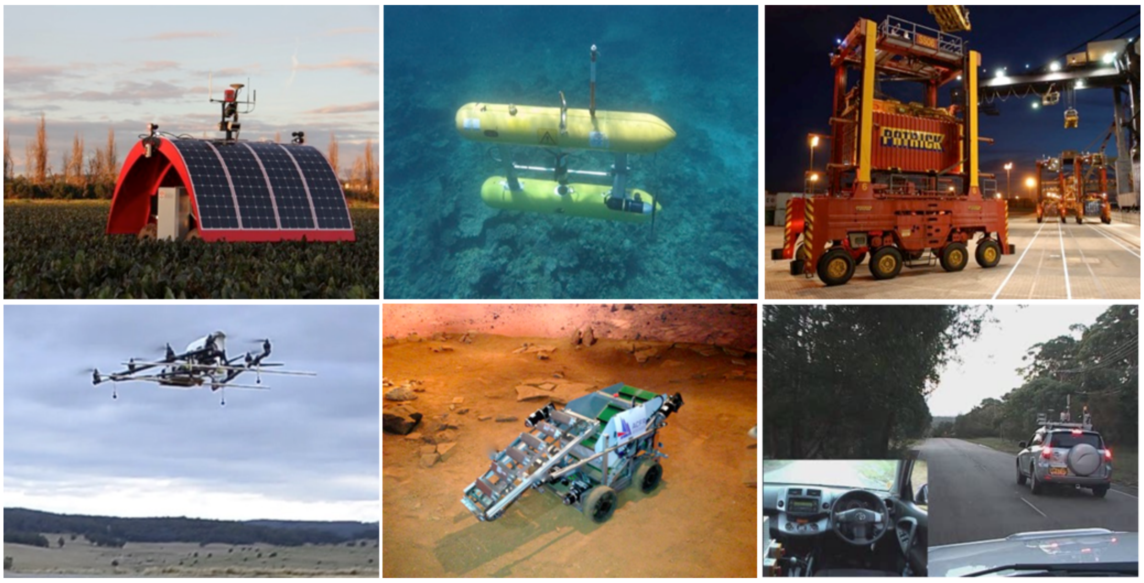


Figure 1.1: Examples of robots operating in the real world doing work. These robots are out in the field harvesting [22], exploring ocean floors [23], transporting cargo [24], taking aerial observations, mining and autonomously driving.

While robotic systems continue to advance and improve, however, there are physical limits that determine how much a single robot can achieve in a given time. For longer and/or larger scale problems, teams of robots will be needed to solve them. This necessitates using multiple robots, and this sub-field is known as multi-robotics[20, 21]. This field of robotics focuses on the additional complexities of considering how multiple robots can collaborate and communicate to solve typically larger or more complex tasks. Traditional solutions to single-robot problems generally cannot solve difficult problems requiring multi-robot systems (without modification) due to not having been designed with these additional concerns that do not exist in the single-robot problem domain.

Some examples of large-scale problems that are not feasible for single robots but multi-robots can solve are: thousands of Kiva robots [25] operating in Amazon warehouses, mapping ocean sea beds [26], rapidly spraying weeds on broad acre farms [27](as demonstrated in our previous work), teams of automated straddle carriers [28] operating at ports around the world, and fleets of self-driving cars from Waymo [29]. To enable the robots to solve these tasks, they must be capable of sensing their environment, performing a set of useful actions, and using this information to plan what actions to take to complete an objective. In this work, we assume that others have solved the sensing and acting parts of robotics and focus on planning and decision-making; in particular, we are interested in planning problems for multi-robot systems.

Planning problems are ones where a problem definition, a list of valid actions or choices, and a particular goal are given; the system needs to find a sequence of choices that achieves this goal or maximises some measure of utility. This class of problems appear in all areas of society and is not specific to science or robotics. Selecting an ideal location to build a hospital [30], finding a route between cities for delivery [31], and selecting which items can fit in which box [32] are all examples of planning problems. The algorithms used to solve them are general and can appear in robotics. For a single robot or a team of robots to achieve anything useful, they need to have a way of planning their actions that lets them achieve their goal and perform practical work. Without planning, robots are simple machines that follow a fixed routine of actions. Robotics versions of the problems mentioned above are choosing where to take measurements to cover an area [33], path planning a route between places to perform tasks [34], and bin packing for autonomous delivery [25]. There are many planning problems in robotics. Specifically, we are interested in *co-ordination* problems. These problems require teams of robots to jointly try and solve an objective in a coordinated fashion, i.e. their actions, when considered together, result in solving the problem. Traditionally, these problems need to be well-defined and specified for a particular instance. However, useful real-world multi-robot problems tend to be long in duration, large in spatial scale, have long sequences of tasks, can involve humans and have ill-defined objectives. Additionally, they can work alongside humans or interact with the information given by humans, which tends to be imprecise and sporadic, and goals can change. These conditions lead to optimising for the current specific tasks and the cost to execute an arbitrary sequence of future tasks despite all the uncertainty.

Similar difficulties can be seen in the recent trends in the related field of automation; there is a modern push to upgrade legacy systems to Industry 4.0 and 5.0 [13, 35]. The core concepts of 4.0 are the integration of cyber-physical systems (robots) and being data-driven. Industry 5.0 aims to improve industry beyond just being automated to now consider their impact on the wider society with the goal of making things more sustainable, resilient and human-centric. These goals of making resilient systems where humans work alongside robots mean that the logistics, manufacturing, and construction industries need to become more flexible, robust, on-demand, and preemptive. Achieving these desired properties necessitates giving up on the previous level of control and predictability that most industry used to have. These human-driven, less controlled, uncertain, large-scale problems with multiple robotic (or human) agents align very closely with the types of problems being addressed in modern multi-robotics research. This overlap motivates examining if there can be any cross-pollination of ideas between the two fields. A typical problem that needs to be solved for industry is the question of resource allocation.

This optimisation of resource allocation is a heavily studied field of operational and automation research called scheduling problems. Scheduling problems are a class of optimisation problems that involve allocating resources to tasks over time to maximise some objective while subject to constraints. These problems arise in various contexts, including manufacturing [36], transportation [37], healthcare [38], and project management [39, 40]. Practical, real-world scheduling problems also have multiple agents and similar conditions to practical multi-robot problems, such as the multiple sources of uncertainty. These sources of uncertainty can be things such as not knowing the future demand for a product, uncertain amounts of time a human can take to complete a task, or accounting for downtime to maintenance [41].

A key insight of our work is that some multi-robot planning problems, particularly task allocation problems, can be modelled as scheduling problems with various sources of uncertainty. These problems have the same underlying structure of assigning resources to tasks over time, sometimes with sources of uncertainty. However, compared to multi-robotic problems, scheduling problems tend to have a clear structure, and they often do not have the same physical considerations that robotic systems operating in a physical environment have. These changes limit the direct applicability of most existing scheduling methods without redesigning for the new operating environment. Despite this, we believe there is still value gained by modelling multi-robot problems as a scheduling problem; through the modelling process, the source of uncertainty and the dimensionality of the problem becomes more apparent. This modelling lens allows the identification of where in the structure of the problem the uncertainty occurs and identifies what additional information is required to make the problem feasible to solve. Additionally, there is potential to leverage existing methods and research from both fields to solve problems.

Specifically, this thesis aims to develop principled and practical planning algorithms capable of addressing the challenges of multi-robot planning with varying sources of uncertainty utilising the lens of stochastic scheduling. Then, we use multi-robot algorithms to develop methods that leverage this insight to solve the complex problem. The algorithms we developed help build understanding around this class of problems and are practically applicable to their motivating problems. This work is not exhaustive but acts as an initial study into this process of approaching and viewing a class of multi-robotic problems as scheduling with uncertainty problems and the potential benefits to the field.

In this chapter, we motivate introduce the concepts that are used throughout the rest of the thesis. We define what scheduling problems are and outline some existing ways they are formulated. Then, we extend this definition to include stochastic elements and highlight some example problems that have this structure. Next, we discuss the class of multi-robot planning problems that this work is addressing. Subsequently, we outline the intersection between these two schools of thought and present some models for real-world problems to which this methodology has applied. Finally, we provide an outline of the main contributions of this thesis.

1.1 Scheduling problems

Scheduling problems are seen across many different domains. The crux of scheduling problems is allocating resources over time in order to complete tasks according to some goal, subject to some constraints. Many scheduling problems exist as part of our society, such as employee shift scheduling, train timetabling, project scheduling, aircraft maintenance scheduling, and even computer operating systems with a scheduler.

A toy demonstrative example of the process of scheduling is a bakery with a single baker who needs to select the order in which they produce their goods. In this example, the bakery can make three types of pastry: cookies, muffins, and brownies. Each pastry has a different amount of preparation time and requires a different amount of time to bake. The bakery has a single oven, so only a single pastry can be baked at a time, but preparation can happen while the oven is cooking. The objective is to find a schedule of preparation and baking operations that minimises the amount of time to bake one of each pastry. Once an item is prepared, it can be left in a waiting area until it is loaded into the oven, which takes no time. In this problem, cookies take 10 minutes to prepare and 30 minutes to bake. Muffins take 10 minutes to prepare but 30 minutes to bake. Finally, brownies take 20 minutes to prepare and 5 minutes to bake.

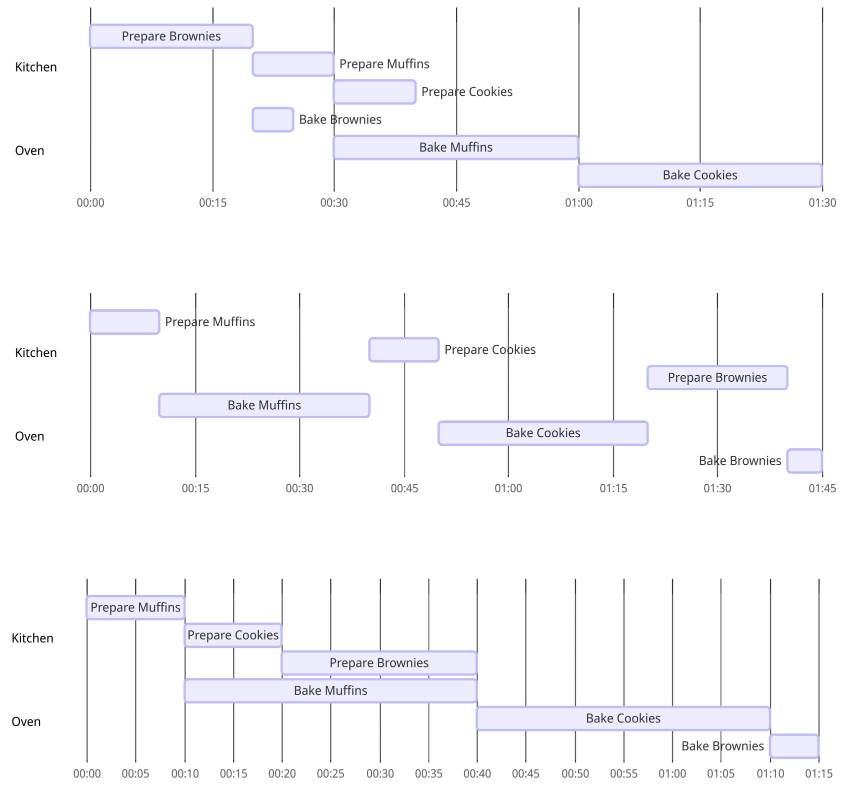


Figure 1.2: Three example schedules with increasing optimality for the simple bakery scheduling problem. The goal is to prepare and bake one of each good in the shortest overall time.

Some possible orderings of the actions the baker can take are shown in Figure 1.2 It can be seen that preparing pastries while the oven is working can reduce the overall time. For this problem, the optimal schedule is to start with preparing muffins (0-10 minutes), put muffins in the oven, start preparing cookies and brownies (both can be done before the muffins finish baking), and finally bake the brownies then the cookies. The total time is 10+30+30+5 for a total time of 75 minutes.

This toy example can be seen as an example of a typical *job-shop* scheduling problem where there are two machines, an oven and a baker, and each job (the types of pastries) needs to pass through an operation with the baker (the preparation) and then the oven to be completed. Real-world versions can have much more complicated process chains and requirements, but this example still stands as indicative of the sort of thinking that goes into solving this class of problems. It also demonstrates the combinatorial nature of scheduling problems. In the simple two-machine three-job bakery problem, the number of possible orders of operations the baker can do is the same as looking at how many arrangements of size six can be made from a set of six items (each pastry has two operations). More formally, these arrangements are known as permutations and would be written as nPr where n is the number of things to arrange, and r is the amount in each arrangement, which is computed using nPr = n!/(n-r)!. From this, it can be shown that 720 possible sequences of actions can happen, but not all of them make sense, such as baking cookies before they are prepared. If you restrict the sequences to only allow valid operations, the number is smaller, but even though this toy example is small, it is important to notice that it is combinatorial with the number of jobs, machines, and operations. For illustrative purposes, imagine you had a bakery with 30 different items(not unreasonable if you think about different types of pastries, fillings, and types of doughs) and wanted to schedule 15 of them to be baked, the number of possible combinations for the baking order (30P15) is larger than the number of grains of sand on earth. Scheduling problems suffer from *combinatorial explosion*, making enumerative methods quickly infeasible. Hence, this is why real-world solutions are smarter about how they perform the optimisation by using heuristics or constraint-based methods; it is possible to find solutions (possibly even optimal ones) for these combinatorial problems without considering every scenario. To first be able to approach the problem and understand how to solve it, you need to be able to define the problem formulation.

1.1.1 Key components of scheduling problems

There are many forms of scheduling problems, but typically, they are defined by the tasks or jobs, the objective function they are optimising against, restrictions or constraints, and the properties of the resources or machines over which the allocation is happening.

In the literature, there is a standard three-field notation used for scheduling problems, $\alpha|\beta|\gamma$ with the three fields containing words that describe the properties of the scheduling problem [42, 43]. The α field describes the machine environment with examples such as single machines, parallel machines, if jobs must be processed on machines in a certain order, and so on. The β field describes the job constraints and rules. Some forms of typical job constraints are allowing preemption, precedence constraints, if there is setup time, or if a job is blocking. Lastly, the γ field is the objective that is being optimised against; for most scheduling problems, this is the makespan of the jobs, but other commonly seen examples are maximum lateness, total weighted completion time and total weighted tardiness. The notation used throughout the literature is not always consistent but tends to approximate or extend this concept of defining an environment/jobs, constraints, and the function that is being optimised.

For this work, we build upon this idea of representation. However, for systems that involve physical systems moving around and cooperating, the standard notation can be insufficient for modelling certain interactions. Such as the cost of moving components from A to B using a robot because this cost is dependent on where the robot is at time T or the constraint that an agent cannot leave a certain zone or travel at unsafe speeds. These sorts of constraints are difficult to model in the existing notation but are easier to capture when using this modification and are closer to a typical robotics planning formulation, which helps to align the fields. Hence, we split the α (machine environment) and β (job properties) into three categories: tasks, constraints, and resources.

Using this modified formulation, the main components of a scheduling problem can be divided into four properties:
1. Tasks: A scheduling problem involves tasks that must be completed within a certain time frame. Each task can have a duration, a start time, and a deadline. The set of tasks may be divided into subsets based on various criteria such as precedence relationships, resource requirements, and task dependencies.
2. Resources: A scheduling problem typically involves a set of resources that are required to complete the tasks. These resources may include machines, workers, vehicles, or any other resource that is necessary to perform the tasks. Each resource may have different capacities and availability times and may be subject to constraints such as minimum and maximum usage times.
3. Objectives: The goal of a scheduling problem is to optimize one or more objectives, such as minimizing the total completion time, maximizing the number of tasks completed, minimizing the use of resources, or minimizing the number of late tasks. The choice of objective(s) depends on the specific application and the decision-maker's preferences.
4. Constraints: A scheduling problem may have various constraints limiting which solutions are feasible. These constraints may include precedence constraints (e.g., a task can only start after another task is completed), resource constraints (e.g., a machine can only be used by one task at a time), time windows (e.g., a task must be completed within a specific time frame), and other operational constraints (e.g., setup times, downtime, maximum travel speed, etc.).

Solving scheduling problems typically involves developing mathematical models that capture these components and using algorithms to find optimal or near-optimal solutions. This process can be challenging due to the combinatorial nature of scheduling problems, the constraints and objectives' complexity, and the problem instances' size. Despite this, significant progress has been made in solving scheduling problems in various applications due to advances in optimisation algorithms and computing power. However, the combinatorial nature of scheduling means scheduling problems are difficult to solve optimally, even for fully observable and known systems. Unfortunately, many real-world problems involve large-temporal–scale, high-dimensional systems and include aspects of uncertainty in various ways. The following are examples of where uncertainty can be involved with each of the properties of a scheduling problem.

Examples of stochastic tasks:
Task durations [44]: The time required to complete tasks can be uncertain and vary depending on factors such as robot performance, environmental conditions, or task complexity. This can be modelled as a probability distribution.
Task arrival times [45, 46]: New tasks can arrive at random intervals, which can be modelled using a Poisson process or other arrival process models.
Task dependencies [47]: Uncertain task dependencies can arise due to unforeseen interactions between tasks or unpredictable changes in the environment. These dependencies can be represented as probabilistic graphs.

Examples of stochastic resources:
Robot availability [48]: The availability of robots can be uncertain due to failures, maintenance, or other factors. This can be modelled as a random variable or using reliability models.
Uncertain resource capacities [49]: The capabilities of resources, such as energy, processing power, or storage, can be uncertain and vary over time. This can be represented using stochastic processes or dynamic resource models.

Examples of stochastic objectives:
Minimizing expected makespan [50]: The objective could be to minimize the expected total time required to complete all tasks, considering the uncertainty in task durations and arrival times.
Maximizing expected throughput [51]: The goal could be to maximize the expected number of tasks completed within a given time window, considering the stochastic nature of task arrival times and duration.

Balancing expected workload [52]: The objective could be to balance the anticipated workload among robots, considering the uncertainty in task requirements and the robots' capabilities.

Examples of stochastic constraints:
Deadline uncertainty [53]: Task deadlines can be uncertain and vary due to changing requirements or other factors. This can be represented using probabilistic constraints or fuzzy deadlines.
Resource allocation uncertainty [54]: The allocation of resources to tasks can be uncertain due to variations in resource capacities, robot availability, or other factors. This can be modelled using stochastic resource allocation models or chance-constrained optimization approaches.

This additional complexity of scale, dimensionality and uncertainty limits the effectiveness of many of the more traditional mathematical constraint-based methods. The addition of uncertainty especially requires a more considered approach to how these problems need to be handled, which is one of the key concepts behind the methodology of the works in this thesis. The types of real-world problems we are interested in addressing can have stochastic elements in a variety of different areas.

1.2 Multi-robot planning problems

Multi-robot planning covers a diverse range of research areas. Within this broad field, this work focuses primarily on addressing the complexities of multi-robot task allocation (MRTA), an important component of enabling multiple robots to solve tasks.

MRTA is a well-studied area of research with several possible taxonomies. To provide context for our work, we use the taxonomy proposed by Gerkey [55], which categorizes MRTA problems along three axes: robots, tasks, and assignments. Robots can be single-task robots (ST) or multi-task robots (MT), which determines if the robots can complete one task at a time or multiple tasks simultaneously. Similarly, tasks can be single-robot tasks (SR) or multi-robot tasks (MR), depending on whether a single robot can complete a task or if some tasks require collaboration between robots. Finally, the assignment can be instantaneous (IA), where all tasks are known upfront and must be assigned, or time-extended (TA), where tasks arrive over time and must be handled dynamically. This taxonomy allows us to describe problems concisely using robot-task-assignment tuples (e.g. ST-MR-TA, MT-SR-IA). This is useful because they are a domain-independent description of a problem, simplifying comparison and improving applicability across various multi-robot scenarios.

In this work, we have primarily studied the ST-SR-IA/TA class of MRTA problems. This means that each task can be completed by a single robot (SR), each robot completes a single task at a time (ST), and tasks are either fully known a priori (IA) or arrive throughout the operation (TA).

A demonstrative toy example of an ST-SR-IA MRTA problem is a smart warehouse with three robots: R1, R2, and R3. Three packages, each in a different location, must be picked up and moved to a delivery location. Each robot can carry one package at a time and can collect and transport it to the delivery location. However, they travel at different speeds R1 travels at 2m/s, R2 travels at 5m/s, and R3 travels at 1m/s. The goal is to transport all packages as quickly as possible. To minimise the total working time, the location of each robot, the position of each package, and travel speed must all be considered. From a multi-robotics perspective, an illustrative diagram of this problem can be seen on the left side of Figure 1.3 with the right showing one possible schedule that delivers all the packages.

This problem demonstrates the similarities and differences between a scheduling problem and an MRTA problem. On the surface, MRTA allocates resources to tasks over time, but it usually has the added complexities of things moving and operating in a physical environment. Examples of properties considered in multi-robotics but typically overlooked in the scheduling methods are communication costs, motion planning, sensing models, communication architecture (centralised or decentralised), and so on. Robotics tends to consider more of the system complexity for the agents rather than assuming moving points on a plane. There are scheduling formulations that also include some of these physical considerations and plan for multiple agents so the fields are not entirely disjoint, but there is limited work that considers properties that are important in real-world field multi-robotics.

1.2.1 Desired properties of multi-robot stochastic schedulers

There are multiple schools of thought and methods for solving MRTA problems, which we will briefly discuss here: The first school is exact methods that try to find an optimal solution (if one exists) for a problem. One of the most common methods is solving constraint-based formulations

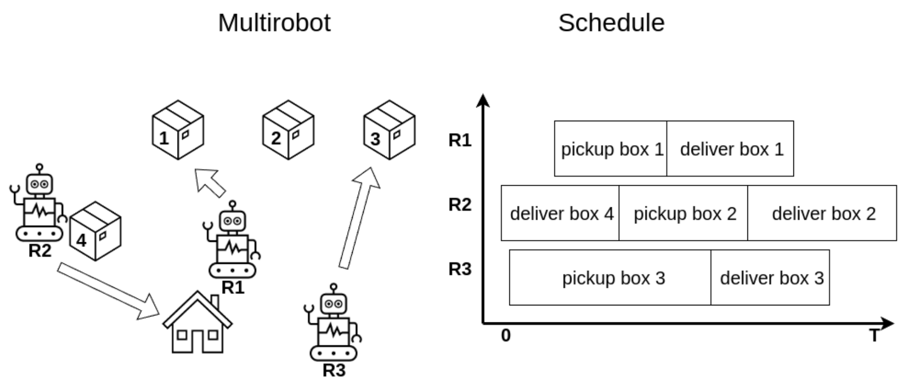


Figure 1.3: Illustrative figure showing how MRTA and scheduling share a structure.

such as Mixed Integer Linear Programs(MILPS) [56, 57]. Constraint-based methods can capture many complex constraints and, depending on the solver, can be relaxed to find partial solutions quickly. Their main drawbacks are that it can be difficult to model complex relationships between variables, that some problems do not scale well in these forms, and that they typically work best for problems that can be fully known and described. Another popular exact method is the class of traditional assignment methods, such as the Hungarian method [58]. Similar to mathematical constraints, they require full information, modelling multiple decisions quickly becomes infeasible to solve, and they are difficult to decentralise. An alternative to solving everything upfront is to handle things as they arrive. Handling things online can be achieved through iterative assignment methods like distributed sequential greedy [59] or auction-based methods [60, 61]. Sequential greedy-based allocations work by using a simple heuristic to measure the value of the task and allocating them in a maximum first manner. These simple algorithms can work well if the problem has the matroid property and the reward function is submodular [62], but many tasks are not; thus, they have an unbounded downside. An alternative to greedily allocating tasks is to let the agents bid on each task and perform an auction to match the task to the agent. However, it can be difficult to model an appropriate bid for complex problems, and they are myopic as they focus only on locally optimal solutions. Finally, there has been a recent trend in applying deep learning-based methods to planning problems [63, 64] where the goal is to take a large amount of data and learn a function that can rapidly generate plans. However, deep learning methods are sensitive to the deviations between their training and real-world scenarios.

The scheduling problems explored in this work involve complex, dynamic environments with various sources of uncertainty, which are hard to model and fully represent in training data. This can limit the effectiveness of deep learning methods, potentially leading to failure or underperformance in novel situations. Techniques such as domain randomization [65], adversarial training [66], transfer learning [67], anomaly detection [68] and active learning [69] can help the model adapt to domain shifts and improve its applicability to these environments [70]. However, while they have been useful in improving the robustness of robot control models or trajectory planning, they have seen limited application in the multi-agent planning domain so far. In addition to this complexity, deep learning methods often struggle with sparse rewards, with long decision horizons, scalability and coordination, which are properties that characterise multi-agent planning problems and pose known challenges [71].

Therefore, deep learning can be useful for specific parts of these problems, such as behaviour prediction [72] or estimating costs[73]. However, its use as a complete end-to-end planner algorithm requires further research and development. Although highly successful hybrid approaches have combined deep learning with a broader planning framework [74], deep learning alone is often not yet sufficiently mature to function as a standalone solution due to these challenges.

Similar to planning, there are many different schools of thought about how to handle uncertainty in the problem that is being solved, which typically break down into two camps: exact or sampling-based, with two sub-camps in each with the goal of optimising for stochastic performance or making things robust. If the uncertainty of the problem can be mathematically defined, exact methods [75] optimise using the exact bounds of the probability. These come with the downside that they require distributions that create the stochastic data to be fully known and modellable. This is not always possible for certain real-world problems, so instead of explicitly modelling the distributions, sampling-based methods sample the environment, simulation or data [76, 77]. Like all sampling-based methods, the quality of your solution depends heavily on the number of samples and the method used to do the sampling. Both of these methods can be used to optimise for some stochastic measure of quality, such as maximising the expected reward or to try and make the solutions robust where every possible scenario is feasible by assuming a maximum bound and ensuring solutions are on the correct side [78].

In this thesis, we are focusing on large-scale multi-agent problems, which means any methods used must deal with these large-scale sequential decision-making problems. The inherent complexity of these problems, having a combinatorial nature and being NP-Hard, makes methods that enumerate unviable due to the computation. Similarly, the chosen methods will need to scale well with the number of tasks, size of the area and number of agents. Additionally, naively handling uncertainty can make methods to produce low-quality solutions or become computationally intractable. We aim to harness predictive methods and intelligent selections of heuristics to make the problems more feasible while trying to limit the loss of optimality these design decisions can cause. In essence, we propose that ideal algorithms for this domain of multi-robot stochastic scheduler should be model-free, scalable, capable of online recomputation, non-Markovian, predictive, non-myopic, and can handle uncertainty. Not all problems require these capabilities in equal measure, but they are typically nice properties for the algorithm to have that can help them handle the complexities of these problems. Real-world multi-robot scheduling challenges are NP-hard, hard-to-define, noisy, combinatorial, expansive, and have interdependencies between multiple components of the problem.

1.2.2 Modelling multi-robot plans as scheduling problems

By looking at multi-robot problems as scheduling problems, we can leverage the benefits of both disciplines' thought processes, algorithms and frameworks.

There are multi-robot problems that initially do not look like scheduling tasks. Still, they can be viewed as allocating resources to tasks over time given some constraints in service of some objective (maximise reward, minimise risk, etc.). This viewpoint allows the practitioner to consider where the problem's complexity lies and what steps and trade-offs can be taken to make the problem feasible.

There are many ways to model and solve problems. We argue that this scheduling lens lets you identify where the complexity and uncertainty are in your problem instance. Then afterwards, with the difficult components identified, you can use whatever algorithm you choose to use to solve them as long as it considers the constraints and uncertainty identified during the modelling. A similar example that is well-known in the multi-robotics community is modelling problems as a Partially Observable Markov Decision Process (POMDP). Subsequently, the goal is to find a decision-making policy that optimises the reward function, which is defined as part of this process. However, solving this process exactly is often computationally intractable, so for practical reasons, people will use an approximate method such as a sampling-based method or heuristic solver. Despite the infeasibility of the original model, the act of modelling and defining the problem in its full complexity lets you understand the constraints and trade-offs needed to make solutions practical and applicable.

1.3 Scope

As described above, the problems we are interested in solving in this thesis cover several research areas. The scope of research areas covered and the intended problem settings will be described in this section.

1.3.1 Multi robot task allocation

General-purpose multi-robot systems should be capable of operating in the real world and must deal with the complexities of doing so. Adding more robots allows the team to complete more work than is feasible with a single robot. Multi-robot planning is a broad area of research. In this thesis, we are focused on the Multi Robot Task Allocation subset of problems. MRTA problems ask, ‘Which robot should we allocate to complete which task’ while trying to complete an overall goal modelled via a reward function.

Many variations of MRTA problems can vary by formulations, such as task dependence, where some tasks cannot be completed until others are done [79]. The tasks that need allocation in problems presented are independent and do not have these types of constraints. Additionally, For this work, we focus on problems where the actions are discrete, such as observing this point or collecting this item. Still, other formulations handle continuous tasks such as information gathering [80]. There are many real-world problems that can be modelled as discrete decision problems or can be discretised. Despite the resolution lost due to discretisation, our methods are still applicable to suitably solve problems since most of the problems we are interested in are nearly impossible to solve optimally. The scale of these problems and the additional complexity of uncertainty necessitate making trade-offs, such as discretisation, to find feasible, practical solutions. Another problem often interleaved with task allocation is the motion planning aspect of completing tasks, known as Task and motion planning (TAMP). TAMP requires the planning over the logical constraints and the physical constraints jointly. The methods presented in this work (besides Chapter 6) consider physical systems operating in the real world to integrate the physical constraints of moving agents around an environment and the costs of these motions while achieving goals. Finally, we are focused on the ST-SR-IA/TA class of MRTA problems. We do not consider problems where a robot can be completing multiple tasks at once or tasks that require multiple robots to complete (such as joint lifting).

Even in this restricted problem format, the types of problems we are interested in are combinatorial and NP-hard. This limitation means that finding optimal solutions for real-world–scaled problems, such as those where you would actually require a team of robots to solve, becomes complicated. Additionally, since we are focused on real-world problems, many come with the added difficulty of uncertainty in various components of the problem. The methods to address uncertainty in the allocation problem require different approaches depending on where and how the uncertainty is included. Furthermore, MRTA problems are still multi-robot planning problems, so they are subject to the typical complexities of multi-robot planners. The key challenges that need to be addressed are the scale of the joint space, communication, and choice of coordination method. One of the main challenges is that in multi-robot problems, the search space grows exponentially with the number of robots. Subsequently, a large portion of the literature is focused on overcoming this hurdle by reducing the computational complexity while still being able to coordinate the team. Similarly, the required amount of communication needs to be considered as the number of agents grows. Finally, the method of coordination needs to be considered; planners can either be centralised or decentralised, and each method has its benefits and drawbacks. For robustness, having a central point of failure is not desired since if one robot fails, the system can continue to operate, and it allows intermittent communication failure not to bring down the whole team.

In this thesis, we want to find methods that can deal with ST-SR-IA/TA MRTA problems that include uncertainty in various components of the allocation problem. Our methods should preemptively account for the future rather than purely react. Furthermore, our method should be scalable to account for real-world–scaled problems with many agents, over large areas, and many tasks.

1.3.2 Scheduling

Since our focus is on a subset of task allocation problems, the methods we present may not be suitable for all scheduling problems. For example, some scheduling problems require multiple-step processes or have task dependencies. These problems are outside the scope of the methods we present currently.

This thesis focuses on solving these allocation problems by modelling the problems as scheduling problems to identify sources of uncertainty and understand the scope and scale. Then, subsequently, this understanding will be used to determine what information can be used to make these problems more feasible.

Additionally, we are focused on larger-scale problems with many agents and tasks. There are existing methods, such as approximate methods like lagrangian relaxation or constraint generation, to find optimal solutions. However, the uncertainty and constraints caused by mobile agents, and the uncertainty limit the applicability of these methods. The intent of this thesis is to work in these classes of scenarios, so the methods we produce need to be suitable.

1.3.3 Planning with uncertainty in various problem components

Real-world multi-robot systems require a nuanced understanding of the type and sources of uncertainty that can be encountered. Traditionally, there are two main categories of uncertainty – epistemic uncertainty, arising from incomplete knowledge of the system or environment, and aleatoric uncertainty, caused by inherent randomness or variability in a system. Epistemic uncertainty can often be reduced through improved sensing, modelling, and learning, while aleatoric uncertainty requires robust or adaptive approaches that can handle inherent variability. Given the complexities of real-world operations, understanding the source and type of uncertainty can help develop capable, efficient, and resilient methods.

In addition to the general forms of uncertainty, the systems we create should be able to deal with different forms of uncertainty from various sources in different parts of the planning problem. The various forms of uncertainty can stem from uncertainty about what tasks the system will need to complete, uncertain environments, stochastic constraints, uncertainty in what resources are available to complete the tasks with, and many other variations.

Systems that can deal with uncertainty in a general sense are desirable, but they are practically infeasible at this point in time to create. The addition of uncertainty into multi-robot planning problems presents several challenges on top of the typical multi-robot challenges mentioned earlier. Most of the work includes state uncertainty or environmental uncertainty, such as in mapping problems. we are interested in looking at how uncertainty in the different components of the MRTA problem changes the methods required to solve them.

Uncertainty is difficult to plan with because instead of having a point of information, you have a distribution of information to consider when selecting actions. The properties of this distribution and the goal of the overall system greatly affect the decision-making process required to select that action. In some problems, it is sufficient to optimise for a summary statistic such as expected value and in others, you are trying to reduce the impact of the worst-performing percentile (risk) of the distribution of outcomes. Additionally, most previous work assumes the unknown parameters of the problems they are solving are drawn from well-behaved distributions and values follow a normal distribution or similar. These models allow mathematical formulations and nice proofs that bound the optimality or prove the effectiveness of their methods. However, the real world is often not well-behaved, so we focus on model-free methods that leverage the data available to build arbitrary distributions. Using these distributions and understanding of the problem's properties, we create forecasts that are used in our methods to deal with the stochastic nature of the problems.

In this thesis, we aim to develop methods capable of exploiting the available information and the underlying structure of the problems to solve large-scale multi-robot planning problems. Furthermore, the methods should account for the complexities of real physical mobile robots operating in complex environments. We focus on maintaining the effectiveness of our solutions while the problem scales in complexity and does not rely on problems following well-formed distributions.

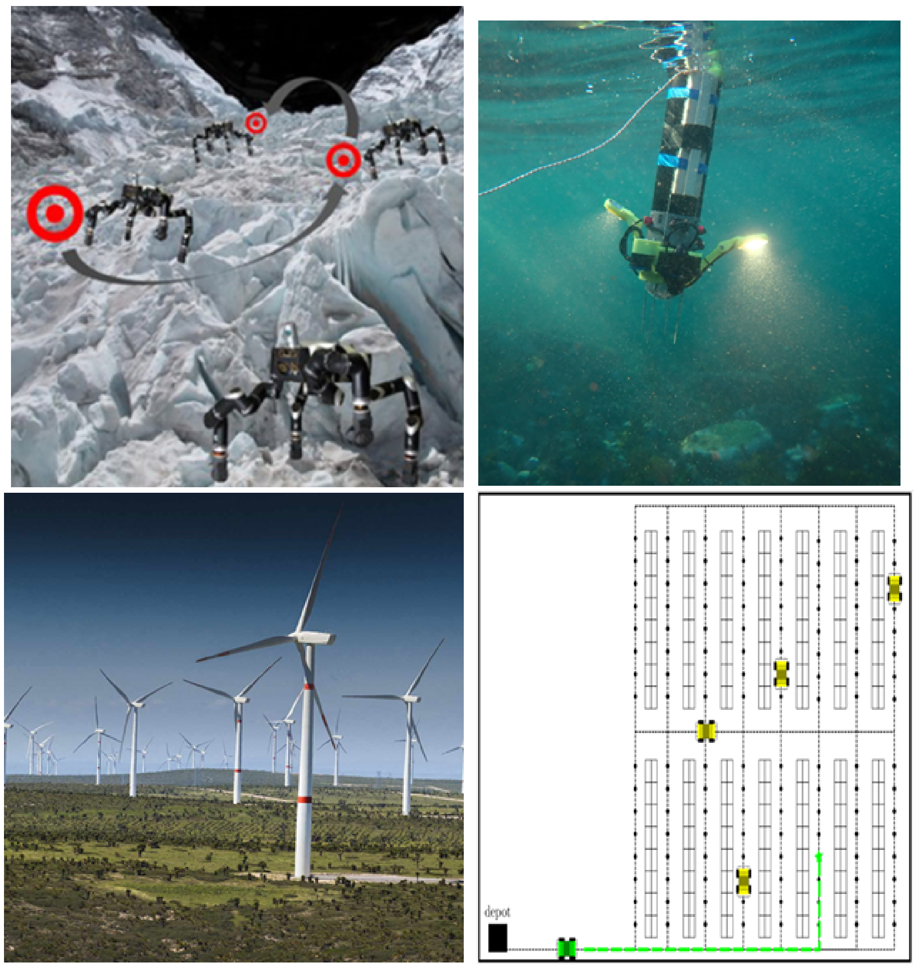


Figure 1.4: Projects addressed in this thesis using our probabilistic multi-robot scheduling framework. (Top Left) Decentralised planetary exploration with map uncertainty, Chapter 3. (Top Right) Multi-agent ocean coverage, Chapter 4. (Bottom Left) Weekly operation for wind-powered hydrogen, Chapter 6. (Bottom Right) Distributed multi-agent warehouse order picking, Chapter 5.

1.3.4 Problem settings

This thesis is concerned with solving real-world practical multi-robot planning challenges in various domains. Through solving this collection of problems, we establish a new theory of robotics scheduling that addresses four fundamental components of how stochasticity manifests. The four components of our probabilistic scheduling framework are stochastic objectives, constraints, tasks, and resources. As part of this research, we model and produce solutions for four practical, real-world problems. Each problem has uncertainty in different components of a typical scheduling problem and needs to be considered differently.

This thesis contributes to a suite of planning algorithms for multi-robot task allocation with our four core types of uncertainty in various scenarios. We emphasise scalable, practical solutions that would be feasible to deploy for real-world-sized problem instances. We consider scheduling with uncertainty in practical robotics and automation applications. This section introduces these problems, which will be explored deeper in the following chapters.

1.3.4.1 Stochastic objectives: intuition-guided multi-robot orienteering (Chapter 3)

This problem deals with stochastic objective scheduling. The objective function determines the utility of a set of actions and defines the planner's goal. In stochastic objectives problems, the tasks are fully known, the constraints do not change, and there are no stochastic consumable resources that need to be considered. The uncertainty lies in the fact that the reward that is received for executing actions is unknown until after the actions have been executed. In these problems, there can be many valid solutions, albeit with an uncertain utility. However, the goal is still to find high-quality plans, so the stochastic value of actions needs to be carefully considered.

We explore the first component of our probabilistic scheduling framework, stochastic objective scheduling, by solving the discoverable edge cost orienteering problem. This problem is a variant of the typical orienteering problem where an agent wants to move around an environment and gather as much reward as possible given a limited budget. In this variation, the cost of traversing around the environment may change from the initial cost, and the agent needs to still try and optimise the expected reward amount. We demonstrate this problem in a planetary exploration context. The problem being addressed is developing a persistently autonomous robotic system for in-situ surface exploration of Europa, a moon of Jupiter. The aim is to enable the robotic system to make decisions and execute tasks independently while minimizing the potential of mission failure due to constraints such as limited communication windows and latency that make teleoperation-based robot control difficult or infeasible.

To solve this problem, the approach being proposed is the development of a multi-agent decentralized Monte Carlo Tree Search planner that enables persistent autonomy. This planning algorithm leverages the capabilities of both the Earth-based ground station and the mobile robot. It enables the mobile robot to formulate its own plan based on its higher definition map of the local environment and is guided by the ground station's most recent long-term plan. The effectiveness of this planning strategy has been demonstrated through the simulation of a planetary exploration problem where the mobile agent has limited computing, and the environmental understanding is initially noisy. This method represents a new paradigm for how to approach multi-agent planning problems.

1.3.4.2 Stochastic constraints: Multi-Vessel-Multi-Float (MVMF) scheduler (Chapter 4)

This problem deals with the second component of our framework, stochastic constraints. Constraints define the rules of the environment and determine if a set of decisions is valid. Stochastic constraints occur in scenarios where the rules of a system change and are driven by external stochastic processes. The tasks that need to be completed are fully known, the objective function is deterministic, and there are no stochastic consumable resources that need to be considered. However, due to the constraints, changing a schedule may be valid in one instance of the constraints, but under a different configuration, the same schedule may be invalid. This invalidation of previously feasible plans is more restrictive than the stochastic objective case, as the uncertainty causes failure (which returns no reward) rather than suboptimality.

As a way to explore the idea of stochastic constraints, this chapter introduces this class of problems in an oceanographic information-gathering scenario. The problem being addressed is the development of a planner for a low-cost multi-agent benthic imaging system used for scientific data gathering of ocean floors, specifically in reef systems. The system uses minimally actuated imaging floats that are deployed in areas of interest to gather data, which are then retrieved by manned or autonomous surface vessels. In this problem, we need to find a joint schedule for a team of vessels operating in the ocean that covers all points of interest using a novel Multi-Vessel Multi-Float system.

The main constraints in this problem include the physical limitations of the system, such as the number and capabilities of the imaging floats and surface vessels. Additionally, there are environmental constraints, such as uncertain current conditions and physical constraints of the ocean floor. The approach being taken to solve this problem is the development of a planner that can optimize float deployments and surface vessel deployment sequences while accounting for uncertain current conditions and physical system constraints. The current version of the planner can solve the deployment planning problem under the assumption that the changes in the ocean currents are slow enough that they can be considered quasi-static.

1.3.4.3 Stochastic tasks: dynamic warehouse order-picking (Chapter 5)

This problem investigates problems with stochastic tasks. Tasks are the pieces of work that the agents complete by performing actions that are subsequently rewarded by the objective function. Stochastic tasks are defined by the uncertainty in the formulation of the task. This task uncertainty can take multiple forms, such as not knowing 'which' tasks need to be done, 'when' the task will be completed, or 'how' to complete the task due to it having stochastic costs.

The problem being addressed is optimising the order-picking process in a multi-agent warehouse. *Order picking* is the process of an agent (robot or person) traversing around the warehouse to collect items from their storage location and then return them to a depot for fulfilment. Orders are not known beforehand and arrive throughout the day of operation, so the planning algorithm must account for this *dynamic* aspect. This dynamic aspect means the tasks that need to be scheduled are stochastic, and any method used needs to deal with the uncertain demands of the future.

Finding optimal algorithmic solutions for the warehouse order-picking problem is infeasible for all but the smallest of scenarios because component subproblems such as routing, batching, and allocation are NP-hard [81]. Additionally, future-specific orders are impossible to predict, so solutions should be capable of accounting for future orders and dealing with the dynamically arriving orders in a timely manner.

We want to find a method to deal with future orders' uncertainty without simply reacting to orders naively. Our method should preemptively account for the future. Furthermore, our method should be scalable to account for large warehouses with many dynamic orders, many agents and many item locations.

1.3.4.4 Stochastic resources: risk constrained scheduling with stochastic forecasts (Chapter 6)

Through solving this problem, we address scheduling problems that have stochastic resources. In a subset of scheduling problems, resources are required to complete tasks and are consumed during the work. The resources are considered stochastic when a stochastic process replenishes them, thus making it uncertain if enough resources will be available when completing a task in the future. The process of obtaining more resources can be uncertain for various reasons, such as unclear production amounts or arrival times.

The problem being solved is the development of a method that can schedule jobs that consume a resource that is produced by a stochastic process. Creating this schedule requires the consideration of when and how much the resource will be available, which depends on the process that generates the resource, which may be difficult or impossible to model. We are interested in optimising the number of completed jobs while minimising the risk that a schedule is invalid due to having insufficient resources.

We demonstrate this problem type in a practical scenario of scheduling green hydrogen consumption operations for the German aerospace center (DLR). DLR wants to schedule robust weekly operations for green Hydrogen that is produced by wind power through electrolysis. For operational reasons, these schedules must be committed to a week in advance and have a known risk of failure.

However, the stochastic resource budget and risk bounds require a method to quantify the risk of a schedule conditioned on the expectation of wind in the coming week. Wind forecasting is a complex temporal problem, so practical solutions must account for the uncertainty rather than working with a summary statistic such as expected valid or bottom quantile. To demonstrate its limitations, consider two schedules that are valid at the end of the week when considering the final capacity. However, when examining every point in time along the schedule, these schedules may be infeasible due to consuming more resources than were available. Scheduling problems are non-Markovian and a time series; hence, the entire history needs to be examined to validate a schedule properly.

1.4 Thesis contributions

The fundamental contribution of this thesis is to view certain multi-robot planning problems as probabilistic scheduling problems with four sources of stochasticity: objectives, tasks, constraints, and resources. This thesis contributes a suite of planning algorithms for several problem domains with various forms of uncertainty. The problem domains addressed in this work are dynamic warehouse order picking, exploring an unknown world with a noisy map, heterogeneous team operation in an ocean environment, and scheduling uncertain resource production based on wind forecasts. We emphasise that the methods and ideas presented here have been tested in these problem domains, but this thesis does not target these specific applications.

The detailed contributions of this work are:

1. The development of a new **Multi-robot scheduling with uncertainty framework**. The novelty is that this is a new way of dealing with real-world and decision-making problems. The theoretical impact should lead to more research by people building upon the ideas presented. The broader impact is it should facilitate more field applications of robotics with a societal and economic impact.

2. A novel **intuition-guided multi-robot orienteering** algorithm to solve the **discoverable edge cost orienteering problem** for a heterogenous team of virtual and physical agents performing information gathering on an unknown world with a noisy map prior which results in a **stochastic objective**. Our method leverages the extensive computation available off-site on a ground station to create a virtual biasing function that acts like an intuition for the ground agent to follow without constraining its decision-making ability based on local sensor information. This method is shown to outperform the ground robot planning on its own and blinding following a plan created by the ground station that does not consider this local information.

3. A novel **hierarchical scheduling framework** for a new **multi-vessel multi-float system**. The presented framework is capable of handling the operational constraints of scheduling the operations between multiple surface vessels picking up and deploying multiple information-gathering floats in an uncertain oceanic environment with **stochastic constraints**. Our

framework has been shown to outperform more generic planners and utilise multiple agents more efficiently.

4. A novel **distributed partitioning based multi-agent warehouse order picking** method that can handle large warehouses with many **stochastic tasks** arriving at unknown points of time in the future and containing uncertain requests that are created according to a stochastic process. Our algorithm builds a measure of expected work by sampling from an arbitrary empirical distribution of historical data. This measure of work is then used to iteratively balance partitions, which has been shown to optimise the expected throughput of orders in heavy loading regimes. Our method is capable of handling scenarios where the orders are entirely dynamic and scales well for large-scale warehouses with orders and many agents.

5. A novel **forecast perturbation based risk constrained scheduling** method that can find risk bounded solutions for scheduling problems with **stochastic resource** levels. This method utilises the expected precision of forecasts to perturb the forecast into a distribution of the expected futures. This approach allows the scheduling of medium-term operations that consume a stochastic resource and is produced by an arbitrarily complex stochastic process.

6. Each method has **analytical results** that show the suitability, scalability and usefulness of each of our developed methods for their designated problems. Despite the computational complexity of the addressed problems, we show that our methods can produce high-quality solutions that are practical and suitable for real-world applications.

7. For each method, we presented **empirical results** from extensive simulation trials. These simulation results validate the theoretical claims and demonstrate the applicability of our algorithms to their designed use cases.

8. We performed **Hardware field trials of the MVMF system** to validate the effectiveness of the MVMF system and the algorithms we present in this work. Our system was found to be practical under the constraints we designed it for. Additionally, the field trials offered valuable insight into future algorithmic improvements that will further increase the system's robustness when operating in oceanic environments.

Parts of the work presented in this thesis have appeared in our previous publications. Specifically, Chapter 4 builds upon our previous work in [82], while the field trial and discussion in Section 4.5

is based on our presentation at the 'Advanced Marine Robotics' workshop [83]. Chapter 4 is based upon and extends on our work in [26]. Finally, Chapter 6 is built upon ideas in our work for a future publication (preprint available [84]). Giovanni D'urso is the primary contributing author in all of these publications.

1.5 Thesis structure

The structure of the rest of the thesis is as follows:

- **Chapter 2** surveys the related work in the areas of optimisation, stochastic optimisation, dynamic vehicle routing, and uncertain cooperative decision-making.
- **Chapter 3** focuses on defining problems with stochastic objectives. We investigate this class of problem by solving the Discoverable Edge Cost Orienteering Problem (DECOP), which is a multi-robot scheduling problem with a stochastic objective. We provide our method for solving the problem by modelling it as a new problem called intuition-guided multi-robot orienteering and then solving it using our sampling-based algorithm. Then, we present an example of this problem and our solution in the context of an autonomous exoplanetary exploration scenario.
- **Chapter 4** defines the problem of scheduling with stochastic constraints. This type of uncertainty is demonstrated through the problem of a heterogeneous team of information-gathering vessels operating in the ocean. This multi-vessel multi-float is a novel solution and an application of a problem with stochastic constraints caused by ocean currents. We present a hierarchical scheduling framework to solve the problem. Next, we validate our algorithm empirically through simulation. Subsequently, we test our simulation results through a field trial on real hardware. Finally, we wrap up with a second smaller field trial examining a future direction of work.
- **Chapter 5** presents the dynamic warehouse order-picking problem as an example of a stochastic task scheduling problem. In this chapter, we first define the warehouse picking problem using a Poisson process that describes the arrivals of tasks and the goal to optimise the steady state throughput of the system. Next, we define a partitioning-based method

and service policy that optimises future orders by balancing expected work across agents. Finally, we demonstrate our solution's effectiveness through simulation.

- **Chapter 6** defines the risk-bounded stochastic resource scheduling problem, which involves scheduling jobs that consume a resource produced by an uncertain process. To address this problem, we create a sampling-based forecast perturbation method based on the expectation of forecast's PPV. These forecasts are then used to create a schedule with a bounded risk. We apply this method to a real-world case study of scheduling operations for a green hydrogen research facility that is powered by wind power. We evaluate the performance of our method through simulation and compare it against other baseline methods. Overall, our method outperformed the baselines and reduced scheduling failures to near zero with limited loss of utility.

- **Chapter 7** concludes this thesis and outlines potential avenues for future work.

Chapter 2

Related Work

Robots and other similar autonomously controlled physical systems can be difficult to model and represent because they are equipped with the ability to observe (sense) their environment, albeit in a noisy, limited way, compute (think) which action to take based on their understanding of their current *state*, and finally execute (act) the selected action. The actions taken can change what it is possible to observe through the interaction between the robot and its environment while simultaneously changing what future actions can be taken. This consideration of these systems' sensing, actuation, computation and physical limitations introduces a wide variety of planning problems that consider these interactions in different ways. For demonstration, we briefly discuss two important planning problems that arise from this interaction *task and motion planning* and *active sensing*, but this is not an exhaustive list. For example, consider the problem of a robot that needs to visit all rooms in a house; the system is equipped with a map of its environment and a model of how low-level motor inputs move the robot around its environment. The solution to this problem is a form of task and motion planning where the plan must jointly consider how the robot's motions move around the environment and the sequence it should do the tasks to achieve the goal. Similarly, consider a related problem where the robot must find an unknown object's location while completing the above task. This additional consideration turns it into a form of active sensing because it must now consider how the selected actions can influence the information/sensing data the robot can obtain.

The methods presented in this work tend to operate on the higher level of the decision-making spectrum rather than the low-level control or actuation side. The planning problems examined here are closer to classical planning or combinatorial optimisation problems. The modelling of the problems and the solution development are motivated and constrained by the physical systems that these models represent while considering how they interact with the higher levels of autonomy. In particular, we are interested in solving *multi-robot* planning problems due to their additional capabilities and benefits compared to singular robots, as explained in Chapter 1. Our focus is this intersection between practical multi-robot applications and complex high-level planning problems that consider the limitations and uncertainty of operating in the real world. In this work, we claim an overlap between the scheduling with uncertainty and the types of multi-robot task allocation problems seen in the real world. This thesis borrows and extends ideas from the fields of multi-robot planning, scheduling algorithms, and combinatorial optimisation problems. The new methods we developed in this thesis help extend the capabilities of these bodies of work. This chapter reviews the existing literature and helps to contextualise the methods presented in this thesis and the importance of their contributions.

We look at existing multi-robot planning algorithms in Section 2.1. We first examine deterministic problems and then extend this to problems with uncertainty involved. In Section 2.2, we examine the use of scheduling algorithms to solve fully known problem instances and then outline existing approaches to deal with problems with stochastic elements. Section 2.3 examines generic combinatorial optimisation algorithms and their use in various planning domains that are relevant to the work presented in this thesis. Finally, Chapter 2.5 summarises the results of our review and identifies gaps in the literature that our presented methods address while also contextualising our contributions to the broader research community.

2.1 Multi-robot planning algorithms

In this section, we review algorithms for multi-robot planning. Multi-robot systems tend to be applied to real-world problems, which often are long duration, large in spatial scale, long sequences of tasks, can involve humans and have ill-defined objectives. Firstly, we discuss how plans for multi-robot systems can be produced for the simpler case when there is no uncertainty. Subsequently, we relax this assumption and review methods for planners that account for forms of uncertainty in the problem or in the execution of a decision-making policy.

2.1.1 Deterministic multi-robot planning

Multi-robot planning is a well-studied field with varying approaches for dealing with the breadth of possible scenarios and system configurations. The key considerations for multi-robot planners that are operating in known controlled environments are the challenges of coordinating the actions of a team of robots, dealing with the potentially exponential growth of the combined action and state spaces of the team, and the typical constraints of robots (or physical agents) that accompany operating in the real-world such as communication, computation limitations, and occupying physical space.

Exact methods

Exact methods try to find an exact optimal solution for a problem if one exists. Common methods for this approach are constraint-based formulations such as Mixed Integer Linear Programs (MILP) or assignment methods. These methods can produce high-quality solutions in small-scale problems, certain classes of constraint formulation or with sufficient computation time. However, they tend to have limited applicability for the large-scale real-world problems we focus on in this work due to the problem size, uncertainty or the online nature.

One common class of multi-robot planning problems is multi-robot coverage [85], where a team of robots is required to jointly take observations of measurements over an area. This problem can be made more complex by considering temporal components of the coverage, such as in the persistent surveillance problem. Mathew et al. [86] deal with the problem of maintaining persistent surveillance while addressing the problem of recharging Unmanned Aerial Vehicles (UAV) with an Unmanned Ground Vehicle (UGV). Using a MILP, they find an allocation of discrete time charging windows and paths that minimise travel costs. While finding an exact solution, it was shown not to scale well to real-world-sized problems and is outperformed by their approximate methods. Similarly, Kaplan et al. [87] used a MILP model for the aerial refilling problem involving multiple vehicles and a single refuelling vehicle. While computationally expensive for problems with more than 12 jobs, this method illustrated the practical utility of framing multi-robot problems as schedules. The computational expense was addressed by using a Simulated Annealing (SA) approach. However, this comes with its own set of limitations, such as not considering queueing, and SA does not produce any metrics on solution quality and can be trapped in local minima.

Some multi-robot task allocation problems can be optimally solved in polynomial time. Assignment algorithms such as the Hungarian algorithm [58] can solve the case where all the tasks are known, and each robot needs to be allocated to a single task. This matching-based type of task assignment was extended by Nam et al. [88] to deal with task assignment problems with resource contention using assignment algorithms. This work is able to find the globally optimal allocations. It allows the matching methods to solve problems with contensions. These methods have the limitation that all tasks need to be known at the start of the planning or a priori. Additionally, assignment algorithms are not suitable for problems where a sequence of decisions that rely on each other need to be made as they only optimise for the current set of tasks.

Heuristic methods

If the problem is too complex or large to solve exactly, concessions must be made to make it feasible within a reasonable amount of computation time. A common approach is to use heuristics to approximate the expected reward for a set of decisions or to simplify the decision-making by using a heuristic to select a decision that will 'tend' to do well under some assumptions about the problem.

Monte Carlo Tree Search (MCTS) is a heuristic search algorithm used for decision-making in complex domains. It operates by iteratively building a search tree, where each node represents a state, and by sampling possible actions and outcomes, it approximates the expected reward for each decision. This sampling is designed to balance the *exploration* of new actions and the *exploitation* of actions with known high rewards. This balancing is done using a modified version of the Upper Confidence Bound (UCB) method for Trees called Upper Confidence Trees (UCT). A survey of recent research on applications of MCTS by Świechowski et al. [89] highlights the growing trend of combining MCTS with other techniques, such as machine learning models to address the complexities of real-world problems. Additionally, it demonstrates the importance of including domain knowledge to improve computation speed or result quality [90, 91] and methods that can adapt their use of MCTS online [92]. Some of the main areas of improvement are integrating human in the loop feedback [93], alternatives to UCT for use as sampling methods [94], and integrating machine learning into the process [95].

For multi-robot problems, MCTS can simulate the decisions for each agent and then evaluate the outcome to find an optimal joint action for the team. By considering the agents jointly, the interactions between agents can be considered, but the scale of the *joint space* is combinatorial according to team size, so this centralised approach is only suitable for small teams. Monte Carlo Tree Search has been extensively adapted for multi-robot systems with a focus on multiple different components of the method or application to problems. Liu et al. [96] enhanced MCTS's scalability through parallelization; there needs to be careful consideration of the running simulations not to break statistical assumptions and maintain the balance between exploration and exploitation. Beard et al. [97] tailored MCTS for adversarial settings by integrating game theory into the algorithm, opting to use Nash equilibrium computations over traditional min-max strategies. This approach allows the modelling of a smart adversarial opponent. However, finding Nash equilibria is not feasible for all scenarios. The influence of simulation policy selection on MCTS's efficacy was explored by James et al. [98], highlighting the role of domain knowledge representation. If sampling is large enough compared to the search space size, there is little benefit over uniform random. Still, if the space is too large to cover an informed sampling policy, even when misaligned, is beneficial. Zerbel et al. [99] presented a rudimentary multi-agent MCTS variant through centralizing computations and sequentially computing the decisions for each agent. This solves much of the complexity of the joint space but limits its applicability for larger search spaces and those with inter-agent interactions. Daneshvaramoli et al. [100] introduced a 'communication-less' multi-robot MCTS algorithm. This method requires all agents to operate in a synchronous manner and have perfect state information of each other and all goals, which makes it too restrictive to apply to complex problems. Another approach to decentralised multi-robot planning is to plan using macro actions rather than discrete actions. Kurzer et al. [101] proposed DeCoH-MCTS; their decentralized cooperative hierarchical planner plans using macro actions, which reduces the problem complexity and has faster convergence. The plan is found using simultaneous games and distributed reinforcement learning. The results are promising for short-term plans with small teams, but it suffers on longer horizon plans due to random sampling.

Greedy strategies employ an often simple heuristic to make locally optimal choices quickly. At each step, the algorithm selects the decision that maximises the current reward or the *greediest* choice. This process continues until a termination criterion is met, such as the budget is exhausted or all agents have an action to execute. This method can be applied to teams of robots by sequentially selecting an action for an agent that maximises the immediate reward [59]. These simple algorithms can work well if the problem has the matroid property and the reward function is submodular [62, 102]. Submodularity is a property of set functions where the marginal gain of adding an element to a smaller set is greater than or equal to the gain of adding that same element to a larger set. More simply, it means there are *diminishing returns*; adding an item to the set later has the same or less value than adding it now, so changing the ordering of the decisions does not increase the value. If the problem is submodular, a greedy algorithm works well because the decisions made at each timestep are near optimal. However, if this property is not true, as it is not for many tasks, they have an unbounded downside. This downside is caused by the optimisation only considering the current state and immediate reward without considering the future.

Applications

Cooperative multi-robot problems can be seen in the Flying Sidekick Traveling Salesman Problem (FSTSP) [103] in this work, they formulated a parcel delivery problem for a van-drone pair. It emphasizes the utility of specific problem formulations for leveraging problem-specific optimisations that can be found in some coordination problems. An important domain of work where multi-robots can be useful is marine robotics. Recent work involves coordination between Autonomous Underwater Vehicles (AUVs) and Autonomous Surface Vessels (ASVs) [104]. This work used a combination of goal allocations and temporal planning to distribute planning across the AUVs while dynamically allocating refuel points for the AUVs to meet the ASV for servicing. While it is able to deal with the temporal aspects of marine robotics, the utility function is limited and cannot handle more complex models. multi-agent spatial load balancing in non-convex environments was introduced in [105, 106]. These works offer insights into allocating spatial regions to different robots, providing potential inspiration for similar scheduling-based multi-robot planning tasks.

Generally, the theme of most multi-robot methods is the balancing act between trying to find an optimal solution to a problem and the realities that force trade-offs. Optimal methods have been shown to work well for some classes of problems or for small instances, but as the scale grows, heuristic or sampling-based methods are more prevalent. These methods come with the downside of sub-optimality, although the downside can be reduced through careful design of the heuristics used or by leveraging problem-specific knowledge.

2.1.2 Multi-robot planning with uncertainty

In this section, we address the work that is more aligned with our work's direction and provide a review of the more general multi-robot planning algorithms that include uncertainty. There are few methods that solve these planning problems exactly as they are NP-Hard, with very large search spaces, and this limits the usefulness of exact methods for all but the smallest problem instances.

Partially observable Markov decision process

Multi-robot planning under uncertainty is a complex problem with widespread practical implications. Uncertainties can stem from partial environmental information or uncertain tasks that need solving. Typically, these uncertainties in the objective function can be modelled as a Partially Observable Markov Decision Process (POMDP) or a decentralized POMDP (Dec-POMDP) for problems solved in a decentralized fashion [107, 108]. A partially observable Markov decision process is an extension of the classic Markov Decision Process (MDP) that is used for decision-making in stochastic environments. The extension to the MDP allows POMDPs to deal with environments where the state is not fully observable. A POMDP is characterised by a tuple of states, actions, transition probabilities, observation probabilities, rewards, and a discount factor. The goal is to find a policy (mapping of states to actions) that maximises the expected reward over time despite the uncertainty in the state information. The state uncertainty is modelled as a belief, which is a probability distribution across the states. Using this belief, the agent's possible states can be computed, and the potential observations and rewards for a given action can be computed. POMDPs are highly expressive and can handle complex problems, although this complexity comes with the cost of computational intractability, which limits their applicability for problems of practical size, which has inspired the development of various approximate methods. The paper by

Kochenderfer [109] discusses the optimization of a shared objective function using DEC-POMDP in multi-agent planning, noting that exact solutions for DEC-POMDPs are NEXP-complete, thus motivating the use of approximate, epsilon-optimality solutions. Furthermore, DEC-POMDPs with communication constraints have also been examined [110], highlighting the trade-off between computation, communication, and performance in multi-robot planning.

2.1.2.1 Modified Monte Carlo tree search

Monte Carlo Tree Search has emerged as a popular algorithm for handling uncertainty, leading to its application in decentralized multi-robot planning scenarios. The work by Best et al. [111] proposes a decentralised extension of MCTS, which solves single robot solutions conditioned on a joint-action space belief. Another variant is introduced by Smith et al. [112], which enhances Dec-MCTS with distributed non-myopic task selection for heterogeneous robotic teams. Li et al. [113] proposed an extension to Dec-MCTS that simulates each agent's tree and introduces factored belief to deal with uncertainty. Despite their promise, these methods are not without their drawbacks; they require all robots to communicate with all others, limiting scalability. Additionally, introducing problem-specific heuristics that leverage underlying structures can improve the generic methods for some scenarios. Another drawback is that dealing with multi-objective functions is not straightforward. The Pareto MCTS introduced by Chen et al. [114] is a multi-objective variation that has shown effective environmental adaptation. However, it is limited by the computational complexity of the Gaussian process used for map belief updating, restricting scalability. Bayesian methods have also been employed to mitigate the computational burden of uncertainty in multi-robot planning. Arora et al. [115] proposed using a Gaussian process (GP) with encoded domain knowledge, which grows linearly with the planning horizon instead of the number of samples.

2.1.2.2 Bayesian methods

Bayesian-based approaches offer a framework for handling multi-robot planning by allowing robots to explicitly represent and update their beliefs about the environment, their actions and the states of other robots. Robots continuously refine their beliefs based on observations through Bayesian inference, enabling decision-making that accounts for partially observable environments and beliefs of others [116]. These techniques are particularly valuable for adaptive systems where agents must optimise their behaviours based on limited or noisy observations or in dynamic or noisy environments.

Popular Bayesian techniques are include Kalman filters[117] (adapted for multi-agent systems in [118]), and particle filters[119], which allow agents to estimate their pose and track positions of other agents or objects in the environment. Similarly, Bayesian optimisation can be used to explore the environment efficiently and is especially effective for problems with expensive objective functions or unknown environmental models[120].

Bayesian uncertainty quantification (UQ) is a principled method to assess the reliability of an optimisation process by quantifying the confidence in the surrogate model predictions; this helps the system to understand what it is certain about and avoid risky decisions in areas it is less certain about [121].

In the context of multi-robot planning, decentralised Bayesian planning [122] allows agents to coordinate actions based on the state of their own beliefs rather than requiring full joint knowledge. Similarly, work by Benevento et al. [123] developed an adaptive multi-agent system that learned a surrogate function that represents an underlying unknown spatial density function for distributed sensing. Bayesian learning has been applied to multi-robot patrolling and distributed sensing problems [124] in a way that allows agents to adapt to the failure of robots.

However, despite their strengths, Bayesian methods are limited by their computational complexity, primarily caused by the cost of updating the posterior distribution for large teams and the size of the state space. Gaussian processes, commonly used in Bayesian optimisation, struggle to handle high dimensional spaces due to cubic cost scaling [125]. Although ongoing research aims to address the cost of Bayesian updates and create approximations [126], this remains a limiting factor that can limit their broader application to the class of problems focused on by this thesis.

2.1.2.3 Other methods

Apart from the aforementioned methods, numerous other heuristic and learning-based approaches exist to address the complexities of multi-robot planning under uncertainty. The Multi-agent informed policy construction (MIPC) proposed by Ma et al. [127] trained policies offline for increasing team sizes and difficulty levels, showing effectiveness in sparse cooperative reward spaces. However, its applicability is limited by the scale of the environment and number of robots. Dynamic tasks and dynamic maps have been addressed using coalition formation for heterogeneous robots with dynamic tasks [128] and autonomous exploration using frontier point selection on a dynamic Voronoi map [129]. These methods represent promising directions for research, offering solutions to specific problems of task assignment and exploration in dynamic environments. A probabilistic approach that combines Gibbs sampling with simulated annealing has also been proposed for multi-robot coordination [130]. However, due to the slow nature of probabilistic approaches, it requires a switch to simulated annealing to escape from local minima, presenting a potential risk of oscillation between modes.

There are a variety of approaches for tackling the complexity of multi-robot plans while considering uncertainty, such as POMDPs, DEC-MCTS, bayesian approaches or specific heuristics. However, similar to deterministic methods, communication, computation, or optimality are still limitations.

2.2 Scheduling algorithms

As introduced in Chapter 1, scheduling problems are a class of optimisation problems where resources are allocated over time. The work in this thesis is related to the concepts of scheduling and, in particular, stochastic scheduling. Fundamentals of scheduling can be found in books by Pinedo [131] with surveys of various forms of scheduling problems by Allahverdi [132, 133]. These surveys address deterministic scheduling problems with setup times and those with *no-wait* constraints. The MRTA is similar to scheduling problems with a sequence-dependent setup cost because the cost for agents moving around to complete tasks is a function of their state, which depends on the last task they have completed. Similarly, some of these problems can be modelled as no-wait because we are interested in optimising for throughput or minimising makespan.

2.2.1 Deterministic scheduling

There are many variants of classic scheduling problems variants [134]. However, these formulations can be NP-Hard and thus hard to solve optimally [135] for many real-world problems. Multi-agent scheduling is similar to the classical scheduling problem of group interval scheduling [136]. This problem requires scheduling a set of k many independent tasks with possibly differing execution times. This problem has exponential complexity, so it has the same scalability problems. One of the important applications of multi-agent deterministic scheduling is the domain of warehouse operations and, in particular, order picking. In order-picking problems, the goal is to allocate orders to agents to collect and return to a depot. This problem is combinatorial (in the number of agents and orders) and has been handled through batching-based methods by batching orders into one tour to reduce the mean travel time. Batches can be found with Ant colony optimisation (ACO) and a waiting heuristic [137], but it does not account for the dynamic aspect of some warehouse problems and only works for fully known deterministic problems.

2.2.2 Stochastic scheduling

Stochastic scheduling is an approach for handling inherent uncertainties that are present in a multitude of real-world problems. The uncertainty can come from various places such as dynamically arriving tasks, job running length uncertainty, failure of machines and so on.

Extending the deterministic order-batching problem to account for the online arriving nature of jobs has been studied in various works, including [138]. In this work, they handle the dynamic aspect through replanning by updating existing plans with new orders. A method to solve the generic scheduling problem with stochastic tasks was developed in [139] and [140] introduced further work that allows scheduling stochastic tasks with stochastic lengths. However, due to the problem's complexity, the scalability of such methods has not been demonstrated. To handle problems where the resources being used during the scheduling are the stochastic element, the resource-constrained scheduling with partially renewable resource problem was introduced in Bottcher et al. [141]. A survey of variants of the resource-constrained project scheduling problem was presented in [142]. The work by [143] extends the resource-constrained scheduling by adding general temporal constraints to the production and consumption of the resource itself. Their method relies on the application of a branch-and-bound algorithm.

Stochastic scheduling problems are seen in the area of renewable energy operation planning; improvements in this field are important to reduce C02 emissions. The major issues are inherent intermittency, variability, and complexity of modelling solar emissions, cloud cover, wind forecasting or the other natural processes used to generate green energy. In [144], the uncertainty related to the amount and timing of renewable energy was dealt with using single-machine scheduling. Bi-objective and Pareto optimal solutions were used to address this uncertainty effectively. Similarly, the problem of day-ahead scheduling of wind and solar microgrids was handled through a MILP [145]. The solution involved minimizing expected operational costs and power losses by generating a set of solar and wind scenarios for the day ahead. However, these methods have limited scalability due to the complexity of their optimisation and thus are limited to short planning horizons.

There has been an attempt to address the complexity by using seasonal ARIMA (SARIMA) to model demand [146] and include a conditional value at risk model. While this work captures the seasonality of wind and incorporates risk into the decision-making, the quality of the ARIMA forecasts is limited, especially for longer horizons, due to wind speed being a non-linear, non-stationary process [147]. The differences between a variety of existing stochastic optimisation methods applied to the power scheduling problem were studied in [78]. They settled on solving the problem using a two-state approach where the first problem finds a robust solution (worst-case) and then is further optimised in the second stage. Their approach does not consider the temporal aspects of scheduling and uses a simple uncertainty model. Ji et al. [148] deals with the allocation of power to uncertain load demands while considering uncertain wind power. The optimization process involved two phases: day-ahead planning and intra-day adjustment. They validate their system on case study data. Another approach to handling the uncertainty is to learn an approximator function using deep learning, such as in work by Wang [149], where they used a reinforcement learning-based method to learn an approximation function for their scheduler for mobile energy networks. The results are interesting; however, the scalability of their method is unclear. Furthermore, learning-based solutions tend to perform poorly on problems with sparse rewards and many hard constraints, such as those in the resource-constrained scheduling problem. The concept of learning an approximator function that can then be used as a heuristic for scheduling in an alpha-go style could be a promising future direction of work for these problems. Adding in more sources of power and storage turns the problem into a multi-objective problem. This problem variant is seen in day-ahead planning for a Virtual Power Plant (VPP) [150]. The solution involved selecting from different technologies, such as solar, wind, fuel cells, and Combined Heat and Power (CHP), and employing Multi-Objective Black Widow Optimization to produce a schedule. Reactive, short-term scheduling as a solution to solar and wind variability has been attempted in [151]. In this work, they estimate the wind and solar over the next hour every 15 minutes using a Weibull PDF, then optimise over that interval. This approach is myopic and unsuitable for problems where the system cannot change its plan frequently. An approach that is similar to reactive is recourse or branching-based methods. In [152], they generate multiple schedules in stages based on branching times. You can find the closest scenario to reality when executing online in the scenario tree. Using these models to plan, data-driven approaches to generate synthetic data forecasts have been attempted. A two-stage planning approach of a single day with subsequent days is tackled in the works by Sperstad [153] and similar works by Wu [154] they use a Markov chain to approximate the distribution from which the data was drawn from. This synthetic function models the distributions for wind and solar, allowing the scheduler to account for their variance in production. Similarly, in work by Liu [155], they use quantile transforms and random Gaussian samples to create an ensemble of synthetic forecasts. All of these methods then use these synthetic distributions as part of a MILP with chance-based constraints. These methods are useful because they do not require estimating wind exactly and fit distributions from the supplied data. The idea of approximating the data and generating synthetic data that represents the potential variance is practical for problems with limited forecast data. The main drawbacks are these methods have only been shown to work for day-head operations, and it is unclear if these modelling methods can accurately capture the complex distribution of wind or solar for longer time scales.

The CPU task scheduling domain is another popular domain with many stochastic scheduling problems. In the CPU scheduling problem, estimating the upcoming tasks or their running lengths is impossible, so the work tends to be on optimising different metrics such as turn-around time, waiting time, or response time. Early scheduling work by [156] looked at dynamic distributed systems with hard real-time constraints where each node is responsible for its own scheduling. The tasks were shared using a bidding routine accounting for periodic tasks and communication constraints using a surplus resource heuristic. If a task is allocated and cannot be scheduled, it gets put up for bidding again, which can lead to potential missed deadlines. Regehr et al. [157] focused on CPU scheduling and introduced the concept of task clusters and task barriers. Later work by Biswas et al. [158] introduced a Bayesian optimization-based novel approach for multi-objective task scheduling in real-time heterogeneous multiprocessor systems.

2.2.2.1 Queuing theory

Online scheduling is a class of scheduling where the jobs or tasks arrive during operation, and it is impossible to know about upcoming future tasks. One method for dealing with the uncertainty of the future is to produce a queueing model of the problem and then use this to analyse the problem. Using standard Kendall notation [159] a queue is defined by the triplet $(A|S|c)$, where its A denotes the *arrival process*, S denotes the *service time distribution*, and the number of servers is denoted by c. This notation has been extended to be $(A|S|c|K|N|D)$ [160] to include queue capacity K, job arrival population N and the job service discipline D. From this queueing formulation, assuming the model is accurate to the real system, it is possible to analyse the arrival rate of jobs and the service rate. Knowing these rates helps compute the required amount of resources to service the expected jobs, estimate how long a job will be kept waiting, and use resources in the system.

Previous works have explored the application of queueing theory to MRTA problems, including coordinating robots by estimating demands [161], vehicle routing problems [46, 162] and integrating queueing and game theory [163].

2.3 Combinatorial optimisation

Combinatorial optimisation problems are a class of problems where the objective is to find the ‘best’ solution according to some function from a finite set of possible solutions, subject to constraints. This best solution is described by a reward function, which can represent maximising profit, minimising cost, minimising distance covered or other such decision criteria. The *combinatorial* aspect of the problem comes from the vast number of possible combinations of decisions that make up each solution. Typically, the problems grow exponentially with the input size and are NP-Hard, which means larger problems are impossible to solve exactly, given our current understanding of computation. Classic examples of these classes of problems are the Travelling Salesman Problem (TSP), Knapsack, and vertex cover. Rather than directly solving these problems, there is a large body of literature on finding heuristics or approximation methods that are as near to optimal as possible in the shortest amount of time. Many robotics problems can be represented as these abstract problems of ‘visit all vertices on a graph then return to the start’ or ‘maximise the reward of packing items into a bag of finite size’, which are the TSP and knapsack problems, respectively. By solving these problems at scale quickly, the solutions can be used as part of the robot’s planning process and typically as part of a larger system that solves a practical, real-world problem, such as delivering packages or monitoring an ocean.

2.3.1 Orienteering

The travelling salesman problem is perhaps one of the most studied combinatorial optimisation problems. In the TSP, an agent is given a graph representing a path between cities, and the salesman needs to visit all cities exactly once and return to the starting point in the shortest total travelled distance. A paper of particular relevance to this thesis is work done by [164], where they solve a TSP over a graph with stochastic edge costs using an And Or Tree policy offline using a prior map from a satellite. This work is similar to the discoverable edge cost problem in Chapter 3. The main differences are the offline computation vs online reactivity, and in our problem the agent has a finite budget to spend per horizon.

Having to visit all possible points and then return to the start can be an overly restrictive constraint for field robotics applications. Some problems have a resource or budget (such as time or energy) that limits the number of observations or places that can be visited. A budgeted variant of the TSP called the *orienteering* problem is a related problem where, given a graph, a budget and the reward associated with each vertex, the objective is to find a sequence of decisions that maximises the total score or reward. Orienteering problems are often used in route planning for logistics, tourism, telecommunications and robotics. Due to its usefulness for practical operations, the orienteering field has seen continuous advancements from its inception to the current day. The foundational work by [165] introduced the original heuristics used to approximately solve orienteering problems, focusing on greedy selection and then heuristic rules to reorder routes to be more optimal. Subsequent work by [166] improves the solving speed by using a Greedy Randomized Adaptive

Search Procedure (GRASP) combined with path relinking to fix broken constraints. In [167], they applied evolutionary algorithms with specific crossover rules based on route regeneration, making it more applicable to orienteering. These methods can solve large problems quickly, but they only work for simple orienteering problems without more complex operational constraints such as time windows. The additional temporal constraints were addressed in [168], where they introduce a fast solution to the Time-Dependant Orienteering problem (TD-OP). This method accounts for variable travel times like those encountered on roads the problem then solves it by modelling it as a MILP. This new stochastic problem is then solved using Ant Colony Optimisation and valid time heuristics. This approach is limited to a single agent and requires full graph knowledge and accurate cost distributions.

The extension of the orienteering problem to the case of multiple tours is known as the Team Orienteering Problem (TOP) [169, 170]. This variation extends the orienteering problem to plan a route for multiple agents to visit places to gather rewards. A TABU-based algorithm with heuristics was used in [171, 172] to drastically speed up computation. Similarly, using a branch and price method, [173] developed a method to find exact solutions for TOPs and TOPs with time windows. For large instances, it can take minutes to solve, which limits their applicability to real-time systems or those with uncertainty. Dealing with uncertainty in orienteering problems has been examined by several authors. In [174], they extend the basic orienteering problem to have stochastic travel and service times while maximising the expected reward. Later works by [175] use an adaptive search to solve the problem and then create potential alternative paths. During online execution, the new information is used to select the best path from the set of expected paths. This approach is online and adapts to new information but is limited because the route ordering is static regardless of new online information.

2.3.2 Vehicle routing problem (VRP)

The Travelling Salesman Problem(TSP), when extended to work with multiple agents while retaining its original constraints, changes into the Vehicle Routing Problem (VRP). The VRP involves determining the optimal route for a fleet of agents to service all demands, which is an NP-Hard challenge [34]. Despite this complexity, there are numerous heuristics that have been developed to make solutions more applicable to real-world problems; a good survey is provided in [176].

Applications that solve VRPs include warehouse order-picking [177], delivering packages with teams of drones [103], and coordinating heterogeneous teams of robots for ocean monitoring [26].

Traditional VRP solutions are static; they have full information and nothing changes, but many realistic scenarios require dealing with new tasks on demand, which makes the problem *dynamic*. Dynamic VRPs [178] account for demands that arrive during operation and their potentially stochastic costs. Usually, demands are distributed by a spatiotemporal process, and the solution is to find a policy that minimises the average time a demand waits to be serviced or maximises the system's throughput. Research into this domain has explored spatially distributed surveillance policies for UAVs [179], Servicing demands with time-windows [180], and information-gathering problems [33].

The complexity of the problem grows as a function of team size, and in order to deal with increasing complexity for larger teams, some methods partition the workspace into smaller sub-problems. The benefit of this approach is that these sub-problems are less complicated to solve than the total DVRP and potentially can be computed in a distributed manner, which improves scalability. For instance, [181] proposed a distributing partition for convex regions, later applying this partitioning using adaptive queueing to solve the DVRP [46]. Similar work on performing decentralised Voronoi coverage in non-convex environments is also shown [105, 106].

In this thesis, the Multi-Vessel Multi-Float problem that is studied in Chapter 4 closely resembles the VRP pick-up and delivery problem variant [182], but a typical VRPPUD problem does not have to consider ocean dynamics, which cause stochastic pickup locations as well as travel times. Recent work in marine robotics that most closely relates to the MVMF problem involves coordination between a team of AUVs and ASVs [104] using temporal logic. The approach relies on defining 'regions' to heuristically score goal allocations. The MVMF problem necessitates a more considered approach to modelling utility since the value of the surface vessel's drop-off and pick-up actions is based on its prior action history. Optimal planning problems for fleets of marine vehicles [183, 184] as instances of multi-robot systems are known to scale exponentially with decision variables. For the modelling and planning paths through ocean currents part of the problem, various methods have been employed, including modelling the ocean in a time-invariant manner [185], in a time-varying way [186, 187], considering uncertainty in flow field estimates [188–190], and for 3D flow fields [191]. Hsieh et al. demonstrated an agent solving an orienteering problem in a flow field [192], which was later extended to work with teams of agents [193]. These methods are not directly applicable because the Multi-Vessel Multi-Float requires different considerations due to the underactuated float hardware and the pick-up drop mechanics of the problem.

Similarly, the warehouse-order-picking problem studied in Chapter 5 is similar to the vehicle routing problem (VRP) [177]. It has been shown that batching orders can improve order-picking performance in tours. Batches for fully known static problems can be found with *Ant colony optimisation* (ACO) [194], linear programming with column generation [195] or learning-based methods [196]. Similarly, dynamic order batching can be handled by and updating existing plans with new orders [138]. However, this does not scale well because of its computational complexity. Order-picking has been modelled as a VRP with time windows by [197] and [198] while also considering delivery vehicle costs. However, their solutions are computationally complex and do not handle dynamic demands. Splicing in orders to existing allocations has been studied [199], but the performance degrades the more dynamic the system. Finally, [200] compares the online picking problem to online job shop scheduling problems, but it breaks down under heavy load cases.

2.4 Uncertainty quantification and sequential decision making frameworks

One of the key contributions of this thesis is the development of a multi-robot planning with uncertainty framework, and the conceptualisation of various forms of uncertainty that can be present in real-world problems. Two related bodies of work are the field of uncertainty quantification UQ and the unified framework for sequential decision making by Powell B. [201].

Uncertainty quantification identifies and distinguishes between aleatoric and epistemic uncertainty, aiming to reduce uncertainty in models and simulations [202]. While UQ offers valuable insights into the impact and behaviour of uncertain systems, UQ often employs computationally expensive methods that may not be suitable for real-time planning and control of large-scale multi-robot systems directly. Future work could explore integrating techniques from the field of UQ within the context of multi-agent planning to enhance the applicability of our multi-robot uncertainty framework. This would involve ensuring realistic assumptions about available data, computation limitations, and developing various uncertainty models for complex systems.

Powell presents a generic framework for understanding and addressing sequential decision-making problems in a domain-agnostic manner. It identifies four main classes of techniques that are used to solve these problems, and presents examples and references from a breadth of literature However, this framework is not specific to the elements of scheduling and multi-agent systems. While the problems presented in this work can be modelled using this framework, doing so loses some of the nuance and ability to identify potential structures that can be exploited to make these problems feasible or improve the performance of more specific methods.

2.5 Summary and current limitations

The existing body of work is broad, and many problems have been attempted to be solved in a variety of ways. Exact methods work well for problems that are small or if a very particular problem structure can be found. Typically, heuristic methods are used because the problems that this thesis is interested in are NP-Hard, combinatorial and uncertain, all of which make finding feasible solutions difficult. Multi-robot planning and scheduling as a field tend to focus on different applications, but there is some cross-over in problems and techniques used in each. Throughout our analysis of the existing research, we found that there did not exist methods in either subfield that are capable of handling the types of uncertainty in the classes of problems studied throughout this thesis, with the desired properties that we believe make them more applicable to real-world scenarios. Properties such as handling large-scale problems, handling complex hard-to-describe uncertainties, handling large teams of robots, and being fast to compute. Previous work has come close, but for each of the problems addressed in this work, we briefly discuss the limitations of the literature and how our methods are different.

The Discoverable Edge Cost Orienteering Problem (DECOP) examined in Chapter 3 is similar to existing stochastic edge cost Orienteering problems. However, our developed method of modelling the team as a multi-robot system leverages computational resources that the existing works do not take advantage of, making it superior for solving problems where the ground agent has limited computational resources. Additionally, by understanding the value of new information, the problem is broken down into several planning horizons that are combined in a non-myopic manner, which is lacking in previous works. They either plan a long horizon, ignoring the value of additional information or commit to short-term decisions with limited consideration for the future.

The multi-vessel multi-float scheduler in Chapter 4 is a new information-gathering problem that the existing methods have not fully solved. The problem has several factors that make it difficult to solve. It is highly combinatorial due to the nature of multi-robot task allocation problems growing with team size. It is heavily constrained because the ocean currents can cause many schedules to be infeasible, given the time windows required for pickup. Operating in ocean dynamics introduces many complications, such as time-varying costs. Finally, the coupling between the drop-off and pickup points must be carefully considered to find a valid solution. The novelty of the problem and the intersection of multiple difficult-to-solve problems means that there are not any existing methods that solve this problem, let alone are able to do it in a scalable manner.

No existing method can solve the warehouse-order-picking problem studied in Chapter 5 in a highly dynamic scenario with fully unknown tasks in a scalable way. The warehouse-specific order-picking solutions focus on different sub-problems, perform poorly for scenarios with high dynamicity or have poor scalability. Similarly, the non-warehouse-focused VRP research does not consider the effects of routing around shelves, returning to the depot to drop off items and warehouse environment spatial distributions of items in warehouses. Routing around shelves can break obstacle-free space assumptions and require non-Euclidean distance metrics. In addition, depot returns introduce a variable cost based on the partition location relative to the depot and the agent's position. Finally, some methods rely on statistical properties about the spatial distributions of demands that may not be valid in a warehouse environment, which can have very skewed order composition distributions. Therefore, there is a lack of practical and scalable methods to solve the warehouse-order-picking problem with highly dynamic orders and high agent utilisation. There are solutions to DVRP problem variants, such as DVRP with capacity, multi-trip DVRP, DVRP in non-convex environments, and DVRP with empirically derived distributions. However, no solution covers all these variations jointly, which is essential to practically solving the warehouse picking problem. This work addresses the specifics of the multi-agent dynamic warehouse order-picking problem in a practical, scalable way.

The problem of 'risk-bounded stochastic resource scheduling problem' and the application of 'wind-powered green hydrogen scheduling' studied in Chapter 6 is similar to renewable energy planning and stochastic resource problems. In this problem, the tasks are fully known, and a resource is consumed and subsequently replenished by a complex stochastic process. The goal is to commit to a week ahead schedule that maximises the reward of the allocated tasks while maintaining an acceptable risk of failure. The stochastic process is complex, so it cannot be modelled through a simple distribution, and because the goal is a schedule or sequence of allocations rather than a single-step decision, much of the renewable energy literature is not completely relevant. The commitment limits the applicability of formulating a decision-making policy as there is no option for recourse as the schedule cannot change.

Overall, each of the methods presented in each chapter introduces something new to the field of stochastic scheduling and multi-robot task allocation. The techniques presented here build upon the ideas of previous research and combine them into new problems with useful novel solutions. Our methods and new problem formulations will be useful for future researchers to build upon further and continue to extend this area of research.

Chapter 3

Stochastic Objectives

This chapter deals with stochastic objective scheduling problems. The objective function determines the utility of a set of actions and defines the planner's goal. In stochastic objectives problems, the tasks are fully known, the constraints do not change, and no stochastic consumable resources need to be considered. The uncertainty lies in the fact that the reward received for executing actions is unknown until after the actions have been executed. In these problems, there can be many valid solutions, albeit with an uncertain utility. However, the goal is still to find high-quality plans, so the stochastic value of actions needs to be carefully considered.

We explore the first component of our probabilistic scheduling framework, stochastic objective scheduling, by solving the discoverable edge cost orienteering problem. This problem is a variant of the typical orienteering problem where an agent wants to move around an environment and gather as much reward as possible. In this variation, the cost of traversing around the environment may change from the initial cost, and the agent still needs to try and optimise the expected reward amount. We develop a method to solve this problem by modelling it as a decentralised heterogeneous multi-agent planning problem. Finally, we empirically evaluate our method through a simulated off-world exploration scenario with limited communication.

3.1 Overview

In this chapter, we consider the problem of scheduling a set of tasks with a stochastic reward. This type of problem is a form of stochastic objective because the amount of utility gained by completing a set of tasks is not known. This type of problem occurs often in practice and has been seen in a variety of problem instances, such as selecting the amount of stock for a warehouse when the amount of sales is unknown [203], choosing the amount of each crop to plant to sell next season [204], or in search and rescue where agents need to explore a mine and find lost miners [205]. In each of these problems, if the rewards for completing tasks were known accurately, the problem could be optimised directly. However, the probabilistic nature of the reward motivates optimising the decision-making to optimise over some probabilistic function instead. Typically, in practice, if a distribution of an action's value can be estimated, most of the methods that address these types of problems focus on optimising for the expectation of reward [206, 207] instead of a specific concrete value. Another popular approach is to minimise the expected worst percentiles of expected reward with approaches such as computing the Conditional Value At Risk(CVAR) [208, 209].

In this work, we are interested in the problem scenario of performing an information-gathering orienteering task over an unknown environment. The specific problem we are trying to solve is an agent traversing around an environment to visit as many Points of Interest (POIs) as possible given a limited time or energy budget. These points represent places operators have deemed the most important to observe. For this problem, we have a prior map of the environment, but the real environment may have changed, and thus the map is noisy. Potential sources of noise for the map are cave-ins, the information was faulty, or the map was given by a satellite with limited mapping resolution. We model this problem as a discoverable edge cost-orienteering problem with a stochastic objective. In this variation, the cost of traversing around the environment may change from the initial expected cost, and the agent must still try to optimise the expected reward amount.

The computational power available on mobile agents constrains the types of planning algorithms that are practical to use in the field. Because of the nature of field operations, mobile agents tend to have limited computational power due to battery and weight constraints. This computational limit restricts the complexity (either scale or planning depth) of the planning algorithms that can be utilised in the field during operations. We posit that mobile robots in these scenarios rarely operate independently, and far more typically, they are part of a system containing a ground station with humans. This ground station can be used to shoulder the computational load by offboarding some of the computation that would normally be on the mobile robot. Offboard systems are much more powerful and thus can run more sophisticated methods. However, they lack the local sensing information available on the mobile platform, so one should not constrain the other. Instead, they should work in tandem, exercising the capabilities of each.

We develop a method to solve this problem by modelling it as a decentralised heterogeneous multi-agent planning problem. The team consists of one physical mobile agent and a *virtual guiding agent*, which acts as an *intuition* for the mobile agent to bias it towards good solutions faster. In decentralized heterogeneous multi-agent exploration scenarios, agents of collaborative, decentralized multi-robot systems need to plan cooperatively to perform their tasks efficiently. This requires effective communication between the agents and the ability to identify states where collaboration is most beneficial. Traditional decentralized coordination approaches often rely on simple communication models that facilitate frequent communication, which becomes infeasible in scenarios with restricted communication capabilities, such as limited bandwidth or communication windows.

Moreover, planning for collaborative tasks is inherently challenging due to the large number of possible states, actions, and sparse rewards that must be considered. This chapter proposes a novel approach that addresses these challenges by incorporating a decentralized Monte Carlo Tree Search (Dec-MCTS) with a modification based on the concept of pheromones taken from Ant Colony Optimization (ACO). Dec-MCTS can be operated with limited communication while maintaining good solution quality [210]. Subsequently, we modify the typical Dec-MCTS by biasing the rollout step of the planner for the mobile agent using ACO. The pheromones used to adjust the biasing for the mobile agent are based on the quality of the solutions found by the virtual guiding agent. This combination allows agents to communicate less frequently while effectively collaborating and optimizing their joint performance without the risk of constraining the mobile robots' decision-making ability.

We demonstrate the effectiveness of our proposed method in a planetary exploration scenario involving a mobile robot with a limited communication window and bandwidth to an Earth-based ground station. The ground station and robot team have a noisy prior of the environment given by a low-resolution satellite image of the operating area. The mobile robot gathers high-quality local information that better understands the cost of traversal, while the ground station has significantly more computational power. Through simulation, we demonstrate that our method outperforms alternative approaches, such as myopic greedy algorithms, traditional Dec-MCTS, and orienteering solvers.

3.1.1 Chapter outline

The remainder of this chapter is organised as follows. Section 3.2 formally defines the discoverable edge cost orienteering problem. Section 3.3 defines the Intuition-guided multi-robot orienteering problem as a variant of the single robot problem formulation. Subsequently, we perform some analysis that proves why a good solution to the multi-robot intuition version is a good solution to the original problem formulation. Following this in Section 3.4 we outline our method to solve the multi-robot intuition problem. Our solution consists of a Dec-MCTS component and the ACO and pheromone-based sampling modifications. Afterwards, in Section 3.5.1, we demonstrate our problem formulation and method to solve a planetary exploration problem. We show through simulation that our method is applicable to this class of problems. Afterwards, in Section 3.6, we summarise the results from this chapter.

3.2 Problem formulation

Problems with stochastic objectives occur in various problem formulations and environments. For this work, we are interested in the problem of a robot performing an orienteering task in an unknown environment. More specifically, we want the robot to visit as many places of interest as possible with a limited time or energy budget. Similarly, the robot has a noisy map of its operating environment that may differ from what will be observed during operation. The question we are addressing is in what order we should visit the observation points and what paths the robot should take in order to maximise the expected number of observations. In this variation, the cost of traversing around the environment may change from the initial expected cost, and the agent must still try to optimise the expected reward amount. To model this problem, we introduce the Discoverable Edge Cost Orienteering Problem (DECOP), which extends the traditional Orienteering Problem by incorporating discoverable edge costs, similar to the Stochastic Shortest Path Problem with Recourse. The noisy prior map gives the original cost, and the physical local environment gives the real cost incurred to visit the point. In the rest of this section, we formally define the problem, briefly discuss methods of solving this problem, and in the following section, we define a new problem whose solution is the solution to the DECOP problem addressed here.

3.2.1 Robot team and environment

Let M be the initial a priori noisy map of the environment $M \in \mathbb{R}^2$ with a list of obstacles O with an unknown ground truth map M^* that can be observed locally during operation. We are given a discrete set of *points-of-interest* (POIs) $v_i \in \mathbf{V} \subset M$ with an associated non-negative reward value $Q_i \in \mathbb{R}$, where subscript i is used to index a specific point of interest. Given the environment map M and POIS $\mathbf{V}$ we construct a connected, undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$. Where the vertices are the points we need to observe, and the edges are the collision-free shortest paths between them. The sub-route between two adjacent positions $e_{ij} = (v_i, v_j), (v_i, v_j) \in V$ is assumed to be the collision-free shortest path, which can be easily found using path planning algorithms such as *probabilistic roadmap* (PRM) [211]. This route is found using the current state of the robots map M. These route costs make up the edge costmap $c_{(i,j)} \in \mathbf{C}$ for each edge $e_{(i,j)} \in \mathbf{E}$ where subscripts (i, j) is used to represent the edge between vertices v_i and v_j. Similarly, there exists an unknown map of the true costs to traverse the edges $\mathbf{C}^*$ that is created from the ground truth map M^*

Using a robot R with an energy budget limit of B that is in the same units that the edge costs. The objective is to find a sequence of ordered edges that make up a path $\mathbf{P} = (e_{(0,1)}, e_{(1,2)}, \dots, e_{(i,p)})$ through the graph that maximises the total collected reward while not exceeding the energy constraint B. If the robot tries to execute a path, it will stop moving when its budget is exceeded, so the number of visited points may be fewer than planned. Additionally, the robot is taking local observations of its environment during its movement and updates its local cost map $\mathbf{C}$ according to its sensor model $\mathcal{O}$.

The robot plans using the costs given by the noisy map; however, the actual cost of an action is given from the ground truth map. For evaluation, we introduce a function that outputs if a planned edge is reachable given the ground truth. The reachable function $\mathcal{R}(e_{(j,i)}, \mathbf{P}_{0 \leq j < i} | \mathbf{C}, B)$ takes the edge that is being queried, the subpath up to that edge and the costmap being used to evaluate reachability and returns a 0 or 1 if it is possible to reach given the budget. The function is computed using

$$\mathcal{R}(e_{(j,i)}, \mathbf{P}_{0 \leq j < i} | \mathbf{C}, B) = \begin{cases} 1 & \text{if } ((\sum_{k=0}^{j-1} c_{(k,k+1)}) + c_{(j,i)}) \leq B \\ 0 & \text{if } ((\sum_{k=0}^{j-1} c_{(k,k+1)}) + c_{(j,i)}) > B. \end{cases} \tag{3.1}$$

Using this reachability function, it is possible to compute the reward that a path would yield if the robot followed the path up until it ran out of budget and the rest was no longer reachable. The reachable reward can be calculated as

$$\mathcal{Q}(\mathbf{P}|\mathbf{C}, B) = \sum_{i=0}^{p-1} (\mathcal{R}(e_{(i,i+1)}, \mathbf{P}_{0 \leq i < i+1} | \mathbf{C}, B) * Q_{(i+1)})), \tag{3.2}$$

where $Q_{(i+1)}$ is the reward for visiting vertex $i + 1$. If the cost map given to this function is the currently estimated cost map $\mathbf{C}$, then it will return the reward for the estimated reachable portion of the path given the budget. Similarly, if the cost map passed in is the ground truth cost map, it will return the portion of the reward that was feasible to reach if the system had perfect global information.

If it was possible to estimate a distribution of costs for each edge in the costmap $\mathbf{C}^*$, then we could assign a probability of each edge being reachable and subsequently compute an expected reward for the entire path

$$\mathbb{E}\left(\mathcal{Q}(\mathbf{P}|\mathbf{C}, B)\right). \tag{3.3}$$

This expectation represents how much reward we expect to gather, given how noisy we expect the difference between the prior map and the ground truth map to be.

3.2.2 Problem statement

A robot R aims to maximize the total reward collected by visiting a subset of the vertices, subject to the energy budget constraints B. Once the robot traverses an edge, it incurs the true traversal cost. The robot can visit a vertex only once, and the reward is collected only on the visitation of a vertex. If the planned sequence of actions includes choices that cannot be reached, only the feasible section of the sequence receives a reward.

Problem 3.1 (Discoverable Edge Cost Orienteering Problem (DECOP))**.** *Given a connected, undirected graph* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$*, a set of rewards* $Q_i \in \mathbf{Q}$ *for each vertex* $v_i \in \mathbf{V}$*. Given initial edge costs* $c_{(i,j)} \in \mathbf{C}$ *for each edge* $e_{(i,j)} \in \mathbf{E}$*. Similarly, the unknown ground truth cost is estimated using* $\mathbf{C}$ *with the real cost* $\mathbf{C}^*$ *being observed during the execution. Using a robot* R *with an energy budget limit of* B *that is in the same units that the edge costs. The objective is to find a sequence of ordered edges that make up a path* $\mathbf{P}^* = (e_{(0,1)}, e_{(1,2)}, \ldots, e_{(i,p)})$ *through the graph that maximises the expected total collected reward while not exceeding the energy constraint* B*.*

$$\begin{aligned} \mathbf{P}^* &= \arg\max_{\mathbf{P}} \mathbb{E}(\mathcal{Q}(\mathbf{P}|\mathbf{C}^*, B)) \\ \textit{subject to} &\sum_{k=0}^{p-1} c_{(k,k+1)} \leq B \end{aligned} \tag{3.4}$$

3.2.3 Stochastic objective

We note that the constraints are based on the system's current understanding of the costmap $\mathbf{C}$, while the rewards are given by how much of the selected path can be traversed according to ground truth costmap $\mathbf{C}^*$. During execution, the robot stops moving when it runs out of budget. This means that paths can become feasible, albeit with a lower-than-planned reward, by ignoring the non-reachable section. The change in reward from a path is based on the discrepancy between the current and the real-world costs. Hence, the reward is really an expectation of how much of the planned path is going to be reachable according to the mismatch between the costmaps.

In order to maximise the expected reward, there needs to be a form of prediction either: about the potential states of the environment(since our objective is based on the unknown environment cost) or the reward of the future actions we can take conditioned on what we currently know.

In this problem, we have a prior model of the environment, and the robot takes observations during execution that update this model. This prior and updates can be used in two ways to approach this problem; we can either leverage prior information to model the environment to predict the distributions of edge costs, which then a probabilistically optimal path is selected, or replan the remaining path when new information becomes available. Leveraging prior information can be beneficial if possible, but we assume it is too hard to estimate an accurate distribution of edge costs for this work due to the complexity of the physical environment. To model these costs, we would need to model the probabilistic distributions of possible physical environments where the initial noisy map data could have been observed. Instead, we start with the noisy sampled map and focus on a replanning-based approach to update the selected paths. However, replanning comes with the additional complexity of selecting a planning horizon and the frequency of when to replan. Naively replanning every time new information is received can be detrimental because the solution becomes myopic and focuses only on optimising local rewards with limited consideration for future prospects, while infrequent replanning loses the benefit of the newly obtained information. Similarly, a short planning horizon maximises local reward, but a long planning horizon has a much larger search space and is harder to compute optimally.

If the understanding of the environment is constantly changing or the expectation varies substantially from reality, then there is limited benefit to using the prior information at all, and the planner/scheduler should only use the newly observed information. However, in this work, we assume that the low-resolution map that gives the cost priors is similar enough to the real cost (with some added noise) that planning a long horizon is still beneficial.

3.3 Intuition-guided multi-robot orienteering

As mentioned in the previous section, we wish to use a replanning-based approach that is capable of exploiting new information gained through the execution of a plan in a principled manner. We want to avoid the myopia of too frequent replanning and still maintain a suitable depth planning horizon. Planning with a long horizon for large-scale problems requires decent computational power. A primary challenge in mobile robot planning is the computational and energy constraints for mobile agents. These limitations restrict the complexity and depth of planning algorithms that can be deployed to real-time and real-world field operations. However, these mobile robots rarely operate in isolation. More often than not, they function as part of a system that contains a ground station manned by human operators to monitor the robot. This system presents an opportunity to treat the system as a multi-agent system. In this system, the ground station will typically have significantly more computational power and can be used to offboard some of the heavier computational load. While the ground station can run more intricate and deeper planning algorithms than the mobile robot, they do not have access to the local real-time sensing data that the mobile robot possesses. Hence, the ideal approach is not to let one system control the other but instead to leverage their complementary strengths. This allows the mobile robot to replan and adapt to the new information while not succumbing to myopia.

In the rest of this section, we first define the Intuition-Guided Multi-robot Orienteering problem (IGMOP), then present some analysis which shows that good solutions to this problem are also good solutions to the DECOP problem.

3.3.1 Virtual agents

The ground station is typically overlooked as a source of computation for decentralised systems because, unlike in centralised systems, it is not the central coordinator. For solutions where the ground station is the central communicator and in the loop constantly, any communication problem can interrupt the mobile robots' operation, which is an undesirable behaviour for most field operations. Instead, we propose a method that allows decentralised systems to leverage the additional offboard computational power while maintaining the decentralisation property. We develop a method to solve the DECOP problem by modelling it as a decentralised heterogeneous multi-agent planning problem and using periodical replanning. In this framework, the team consists of one physical mobile agent and *virtual guiding agents* produced by the ground station. These virtual agents act as an *intuition* that rapidly guides the mobile agent's planner towards good solutions. To incorporate the intuition from the virtual agent, we reframe the DECOP as a collaborative orienteering problem between the mobile robot and the virtual agent called the intuition-guided multi-robot orienteering problem. This problem is similar to DECOP, with the main change being that the reward function incorporates bonus utility for mobile agents following the path suggested by the ground station. Solving this reward is equivalent to finding a joint path between the real and virtual agents. Each agent can utilise their benefits by modelling the problem in this way rather than optimising purely for the ground station or mobile robots' best solution. The ground station can plan with a longer horizon, and the mobile robot can plan more locally to use its limited computing in order to best benefit from its local information while being biased towards these longer-term states given by the ground station's suggestions.

Formally, we define this problem as follows. The ground station simulates a set of A many virtual agents $\mathbf{A}$ with the same starting location and a larger budget than physical agent $B_a > B$. This larger budget allows the virtual agents to plan longer horizon plans than what would be feasible for the physical agent. These virtual agents each plan a solution to DECOP using the same information available to the physical robot, and we denote these virtual agent solutions as $\mathcal{P}'$. Now we define a function that returns the bonus utility (intuition) of the physical agent following edges that are in the suggested paths $\mathcal{P}'$. The intuition function is calculated as follows,

$$\mathcal{I}(\mathbf{P}, \mathbf{P}'|\mathbf{C}^*, B, A)) = \left(\sum_{P' \in \mathbf{P}'} \mathcal{Q}(P \subseteq P'|\mathbf{C}, B) \right) /A. \tag{3.5}$$

The virtual agent paths $\mathcal{P}'$ are planned using a larger budget, but since only the subset between the physical agent's path and the suggested paths is considered, the total cost cannot exceed B. Additionally, since the function is divided by the number of virtual agents A the function returns the proportion of overlapping between the physical and suggestion paths. The strength of the intuition on the overall behaviour of the physical agent can be controlled using a scaling factor α. This parameter needs to be selected so the intuition cost does not dominate the expected reward of the ground station's path and adjusts the system behaviour between favouring either of the two components.

Problem 3.2 (Intuition-Guided Multi-robot Orienteering Problem (IGMOP))**.** *Given a connected, undirected graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$, a set of rewards $Q_i \in \mathbf{Q}$ for each vertex $v_i \in \mathbf{V}$. Given initial edge costs $c_{(i,j)} \in \mathbf{C}$ for each edge $e_{(i,j)} \in \mathbf{E}$. Similarly, the unknown ground truth cost is estimated using $\mathbf{C}$ with the real cost $\mathbf{C}^*$ being observed during the execution. Guiding paths $\mathcal{P}'$ (that are solutions to Problem.3.4) provided by the virtual robots are used to compute an additional reward using the intuition reward function $\mathcal{I}$ Using a robot R with an energy budget limit of B that is in the same units that the edge costs. Using a team of guiding virtual agents with an energy budget limit of $B_a > B$ and an influence weighting parameter α. The objective is to find an optimal sequence of ordered edges that make up a path $\mathbf{P}^* = (e_{(0,1)}, e_{(1,2)}, \dots, e_{(i,p)})$ that maximises sum of the expected total collected reward the weighted intuition utility while not exceeding the energy constraint B.*

$$\begin{aligned} \mathbf{P}^* &= \arg\max_{\mathbf{P}} \mathbb{E}(\mathcal{Q}(\mathbf{P}|\mathbf{C}^*, B)) + \alpha\mathcal{I}(\mathbf{P}, \mathbf{P}'|\mathbf{C}^*, B, A)) \\ \textit{subject to} & \sum_{k=0}^{p-1} c_{(k,k+1)} \leq B \end{aligned} \tag{3.6}$$

3.3.2 Intuition effectiveness analysis

When analysing the effectiveness of the IGMOP as a solution to DECOP, the first obvious question that needs to be asked and answered is, 'Does a good solution to this problem actually yield a good solution to the original problem'? To prove that the solution to the Intuition-Guided Multi-robot Orienteering Problem (IGMOP) is a solution to the Discoverable Edge Cost Orienteering Problem (DECOP), we need to demonstrate that by solving IGMOP, we effectively maximize the objective function of Problem 3.4 as defined in the DECOP.

The main difference between the objective functions in the DECOP 3.4 and IGMOP 3.6 is the addition of a term in Problem 3.6 that accounts for the value of following paths suggested by the ground station, represented by $\mathcal{I}$, and a scaling factor α. Now, let us consider the case when the mobile robots follow the intuition the ground station provided. This would mean that they are taking advantage of the intuition provided by the virtual agents. Since the ground station has a virtually infinite computation budget, the virtual agent paths are more likely to be close to optimal, ensuring that the agents maximise the reward with their budget. The benefit of this plan is based on the understanding of the environment at the time the plans were produced. In scenarios where the mobile robot's plans deviate from the paths suggested by the virtual agents, the marginal value of these new actions should be larger than the benefit of following the intuition. As a result, even when the robots do not strictly follow the intuition, the overall objective function in Problem 3.6 would still be reasonably optimized.

One key aspect to emphasize is that the error introduced by following the virtual agent's intuition is bounded. This is important because if the ground station and physical robot have a mismatch in information, this will cause the intuition from the virtual agents to be of minimal value to the ground station. The mobile robots maintain their agency and can ignore any suggested paths that are detrimental to their performance. Consider the case where the virtual agents suggest paths that are completely disjoint from the paths planned by the physical robot; the value of the intuition function $\mathcal{I}$ would be 0. However, if the physical robot chooses to take paths that share actions, there must be some value for that action and some benefit that is encoded through the intuition function. If the actions were of low value, they would not be in the suggestions from the virtual agents as they are also trying to maximise the same reward function. Due to the limited computation capacity of the robots, the intuition provided by the ground station helps deal with myopia and tends to be more beneficial than detrimental for a certain class of problems.

3.4 Intuition-guided hybrid planner algorithm

In this section, we define our method to solve the DECOP by solving the IGMOP using a decentralised multi-robot planning algorithm. We solve the IGMOP using a decentralized Monte Carlo Tree Search using a team consisting of a physical robot and virtual guiding agents. We introduce a modification for the typical Dec-MCTS for the physical robot that biases the rollout using the virtual agent's suggestions to speed up the convergence between the two. This biasing is computed using the idea of pheromones from ant colony optimisation. This combination allows agents to communicate less frequently while effectively collaborating and optimizing their joint performance without the risk of constraining the mobile robots' decision-making ability.

3.4.1 Decentralised Monte Carlo tree search

The basic MCTS algorithm is a non-myopic sampling-based biased search algorithm that is suitable for large state space problems with non-trivial objective functions. The algorithm starts from an initial condition and then iteratively grows the search tree by selecting the next state to explore from a set of actions to take from the selected state. From the selected state, a sequence of actions is randomly sampled using a heuristic (e.g., biased random sampling) until the termination condition has been met (e.g., length of sequence); this process is called a *rollout*. The score from the rollout is then used to update the tree's expectations, and the process continues.

In more detail, MCTS occurs in four steps that occur in an iterative manner until a termination state occurs. The four steps are:

- *Selection:* Starting from the root node (current position of the robot), use UCT to select a sequence of child nodes until a leaf node is reached.
- *Expansion:* If executing this action does not exceed the robot budget B, the leaf node selected by the selection step is expanded by picking a valid sequential action.
- *Simulation:* Given the history of the selection and newly expanded node a rollout is computed. The rollout typically consists of selecting a random sequence of actions until the budget B is exhausted and the search terminates.
- *Backpropagation:* Using the rollout results and selection the reward for the entire path is computed. This score is then backpropagated up the tree, and the summary statistics for each node on the path are updated.

The decentralised extension to MCTS, namely *decentralised MCTS* (Dec-MCTS) [111], is a modification to the MCTS algorithm that allows multiple agents to coordinate to solve a joint objective function for the whole team. Each agent finds its own solution by growing their own MCTS tree while considering the set of potential actions by other agents. The algorithm allows for finding an asymptotically optimal joint set of solutions for the whole team. This process occurs by agents asynchronously planning and cycling between the three phases:

1. The agent performs a tree search over its own action space while considering the actions of other agents.
2. Compress the search tree into a probability distribution q_n. These distributions are a compressed version of their tree that is the top k many paths the agents would take and their associated probabilities.
3. Broadcast the probability distributions to other robots.

When new messages are received from other agents, this information is included for future rounds of the tree search. This process continues until the computational time limit is reached. For node selection, Dec-MCTS uses a modified version of UCT called discounted UCT (D-UCT), which accounts for including information from other agents' plans that are non-stationary. As the agents plan, the distribution of actions they intend to take will change, and this modified UCT favours the newer information.

The main difference between the original Dec-MCTS algorithm and ours is the addition of pheromones that are computed using the probability distributions. These pheromones are then used to bias the rollout of the agents.

3.4.2 Pheromone based sampling

In this problem, we are trying to get the mobile robot to follow the paths suggested by the virtual agent. Randomly sampling decisions and hoping they overlap is an inefficient use of the limited computational power available. We improve the collaboration between the team of agents by using the idea of pheromones from ant colony optimisation. These pheromones can be used to bias the mobile robots to select paths that the virtual agents have found to be high quality.

Ant colony optimisation is a swarm intelligence biomimetic algorithm where the algorithm is designed to mimic the behaviour of social natural agents, in this case, the coordinated foraging of a colony of ants. ACO has been used as a metaheuristic applied to many combinatorial optimisation problems. It is able to find good quality solutions to these classes of complex problems through the repeated application of a few simple generic rules. A singular ant cannot effectively find food or communicate complex ideas, but collectively, through a shared signalling method, they can bias their overall behaviour towards solving a problem. As an ant travels, they leave a constant amount of pheromone. When a new ant reaches the decision point, it uses the amount of pheromone on the paths to make their decision. They can choose to follow this path or select a new path. If the same path is selected, multiple ants overlay their pheromone, and it becomes stronger, which biases future ants to follow the same path. Additionally, pheromones decay over time, so pheromones on less desirable paths will decay if they are not repeatedly selected during the search process. The selection process is biased towards pheromones but is not restricted by it, which allows the colony to do exploration. If a new action turns out to be beneficial, subsequent ants will choose the new options, which will lead to exploiting the new information.

This concept of ants laying down their pheromones and then biasing the decision-making of future ants is fundamental to our method. In our algorithm, Each candidate path for the virtual agent from the ground station can be considered a virtual ant laying down pheromone trails. The pheromones produced by this virtual swarm are transmitted to the mobile robot, where during its Dec-MCTS search, the pheromones of high-quality solutions will act as an intuition and guide its search. The plans from the ground station are used to modify the sampling likelihood of each feature, and the more times the feature appears in the compressed ground station tree, the stronger the pheromone. This means that during rollouts, the robot will sample features that the ground station has deemed to be important more often and hence speed up convergence to the ground station's plan while maintaining the robot's ability to explore locally.

Pheromone calculation

A reminder that $\mathbf{P}$ and $\mathbf{P}'$ are the paths selected by the physical agent and virtual agents, respectively, and e_ij is the edge between vertex v_i and v_j. To construct the pheromones and update their weights, we use the following process. The initialisation happens when the system first starts planning then, whenever the Dec-MCTS update step happens, the pheromones are updated using the new set of information sent between agents.

To initialise the system we create a pheromone weight τ_{ij} for every edge and set its value as 0, $\tau_{ij} = 0 \forall e_{ij} \in \mathbf{E}$. Upon receiving the set of probability distributions q_n and paths $\mathcal{P}'$, we update the pheromone weights. We first compute the change in pheromone value $\Delta\tau_{ij}^P$ for each guidance path $P \in \mathbf{P}'$. The path pheromone contribution can be calculated using:

$$\Delta\tau_{ij}^P = \begin{cases} q_n^P \cdot \mathcal{Q}(\mathbf{P}|\mathbf{C}, B) & \text{if path } P \text{ contains } e_ij \\ 0 & \text{if path } k \text{ does not contain } e_ij. \end{cases} \tag{3.7}$$

q_n^P is the normalised probability of path P being selected out of the set of candidate paths $\mathbf{P}'$. This process allows the weighting of the pheromones by the quality of the suggested paths as modelled through the compression process in Dec-MCTS. The evaporation step is controlled by evaporation parameter $\rho \in (0-1)$ and controls how much of the previous updates worth of pheromones remain after each step. The pheromone updates and evaporation are then combined together into a global update step, which is calculated as

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{P\in\mathbf{P}} \Delta\tau_{ij}^p. \tag{3.8}$$

This process results in the new pheromone weights that the planner will use until new information arrives at a future time.

Rollout biasing

Given rewards for each action and the pheromones, we want to find a sequence of actions that visit as many POIs as possible. Finding an optimal solution for the DECOP problem is hard due to its combinatorial nature, requiring planning over all possible combinations. Since we are solving the IGMOP, which in turn solves DECOP, we have access to the suggestions from the virtual agents encoded as pheromones τ_{ij}. Using this additional information, we bias the rollout function of the physical agent to speed up convergence towards high-quality solutions. Similarly, since we are optimising coverage, we also want to encourage the selection of actions that have a higher reward-to-cost ratio. The rollout biasing that controls the likelihood that a new action (a selection of moving from one POI vertex to another along the shortest known path) will be selected is computed using:

$$p(e_{ij} \notin \mathbf{P} | \mathbf{P}, \mathbf{P}') = \frac{Q_j}{c_{ij}} + \lambda \cdot \tau_{ij} + \epsilon. \tag{3.9}$$

Where λ is a parameter that controls the strength of the pheromone biasing, and ϵ is a small positive non-zero value that ensures that all possible actions have a non-zero probability of being selected, which is necessary to maintain asymptotic optimality.

3.4.3 Analysis

Finding the joint solution between the virtual agents and the mobile robot would be easier for a centralised planner who directly plans in the joint space. However, The size of the joint space and the operating conditions make these centralised methods infeasible. Dec-MCTS allows the planning of the agents in a distributed manner without needing to consider the joint space. However, Sampling-based algorithms like MCTS inherently bias their search towards the state of the action sequence, making it challenging to discover optimal sequences that involve suboptimal choices, especially in large, sparse multi-robot collaborative problems such as these. In these scenarios, decentralised random sampling of the search space for each agent will require extensive time to find these joint solutions. This is especially true for problems where the mobile agent may have

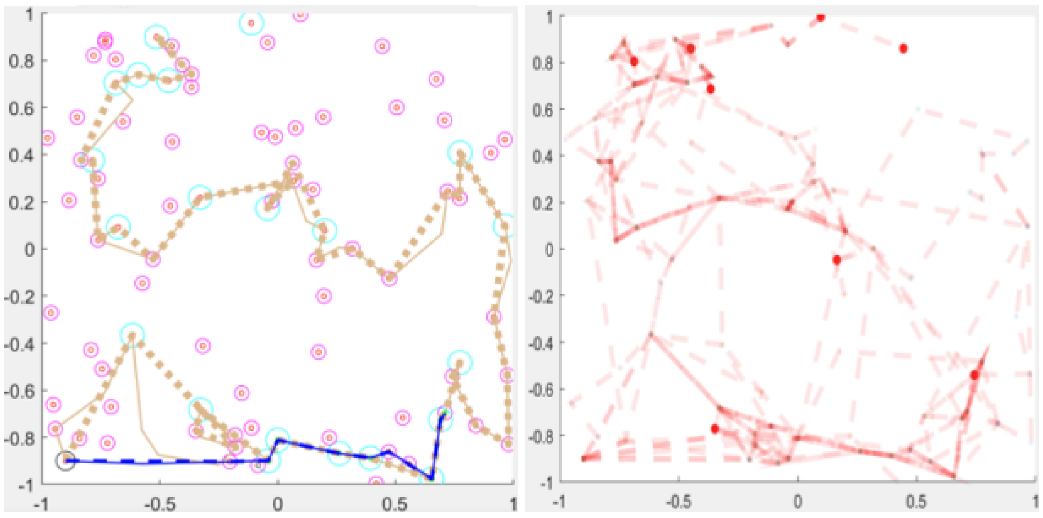


Figure 3.1: An illustrative example of the pheromones after a round of Dec-MCTS planning. (Left) The highest-scoring virtual agent plan computed by the ground station is shown in orange. There are multiple virtual agent plans $\mathbf{P}'$, but they are omitted for clarity. The current plan for the mobile robot is shown in blue. (Right) The pheromones τ_{ij} are computed from the virtual agent plans. The intensity of the edge and vertex colouring shows the pheromone intensity. Darker edges and vertices have been visited more times by virtual agents. Pink circles are worth one reward point, and cyan circles are worth three points of reward.

to take what it thinks is a suboptimal decision but is highly regarded by the ground station. However, because we know that the high-reward solutions are those where the mobile robot follows the virtual agents by integrating the concept of pheromones from Ant Colony Optimization into the rollout process, the modified approach guides the sampling towards faster convergence in these collaborative problems. Similarly, while the addition of intuition pheromones can provide valuable guidance, their application needs to be used intelligently. It is important that the rollout does not strictly adhere to them; care needs to be taken that all possible actions remain selectable in order for the agent to maintain its autonomy. Additionally, this is an important property for the MCTS algorithm, which can only achieve asymptotic optimality if the entire action space can be sampled. If they have been applied correctly and to the appropriate problem, they can use this off-board computation that was not available before. Thus allowing the decentralised solving of this multi-agent planning problem with limited computation while still maintaining the decentralised property and low communication requirements of the Dec-MCTS algorithm.

The performance of the IGMOP algorithm is influenced by the selection of its parameters for the specific problem instances. Three parameters of importance are briefly discussed here: the number of agents $\mathbf{A}$, the intuition weighting factor α and the budget of the agents B_a. Firstly, the number of virtual agents $\mathbf{A}$ can vary based on the system's needs. If an exact optimal solver exists for a problem, then all virtual agents will share the same solution, and additional agents will provide limited additional benefits. However, for larger, more complex problems, some methods, such as sampling-based approaches, can create a diverse set of solutions. While some of these solutions may dominate in optimality according to the ground station's current understanding of the problem states, this diversity of solutions allows additional agents to give a diversity of guidance that can be useful if the updated state information causes deviation from the suggested optimal path of a single virtual agent. Similarly, the virtual agents' budget B_a determines the balance between the computation available at the ground station and the ability to potentially overcome the physical agent's planning myopia. An excessively large budget may compromise the quality of the solutions generated by virtual agents for complex problems with large state spaces. Conversely, setting the budget too close to B restricts the usefulness of the virtual agent's guidance and does not leverage the additional computation available. Lastly, the intuition weighting parameter α governs how strongly the mobile physical agent adheres to the guidance of the intuition provided by the virtual agents. It can be interpreted as a measure of trust between the ground station and the physical robot, which reflects how closely the estimated costmap $\mathbf{C}*$ matches the real costmap $\mathbf{C}$. In scenarios where the costs are perfectly known, then the value of additional new information gathered by the mobile robot is diminished, and it should follow the prescribed paths. However, in the presence of uncertainty, a degree of flexibility allows the system to adapt to environmental observations, with the level of adaptation controlled through the value of α.

The structure encoded by the pheromones can be seen visually in Figure 3.1 where the left is the plans for the agents and the right is the pheromones produced by the team of virtual agents. The cyan circles are worth three times as much reward as the pink circles, and it can be seen that the darker edges or stronger pheromone signals tend to cluster around the areas that have some useful local structure. The bottom right section, middle and top left sections have a strong clustering of rewards, so the pheromones around those areas are heavily deposited. On the contrary, the other areas of the environment have less dense clusters, so the pheromones in those areas are weaker. The connective edges between these areas also have fewer pheromones deposited due to the searching process. There are many possible options that can result in visiting these areas of exploitable structure. This is in contrast to a traditional ACO where the repeated visiting lays down pheromones over the same path repeatedly; by combining the pheromones with the structured biased tree search, the depositing of pheromones follows a different distribution.

3.5 Autonomous exoplanetary exploration

In this section, we evaluate the performance of our intuition-guided hybrid planner algorithm in an autonomous exoplanetary exploration problem. This exploration problem is an example of a Discoverable Edge Cost Orienteering Problem (DECOP). An example instance of the problem and a solution are shown in Figure 3.2. First, we explain the motivation for this problem and define the parameters of our simulation. Subsequently, we perform multiple simulations to empirically demonstrate the ability of our method to solve this problem. These results are used to compare our method against a similar method that addresses the problem directly without adding the guiding virtual agents.

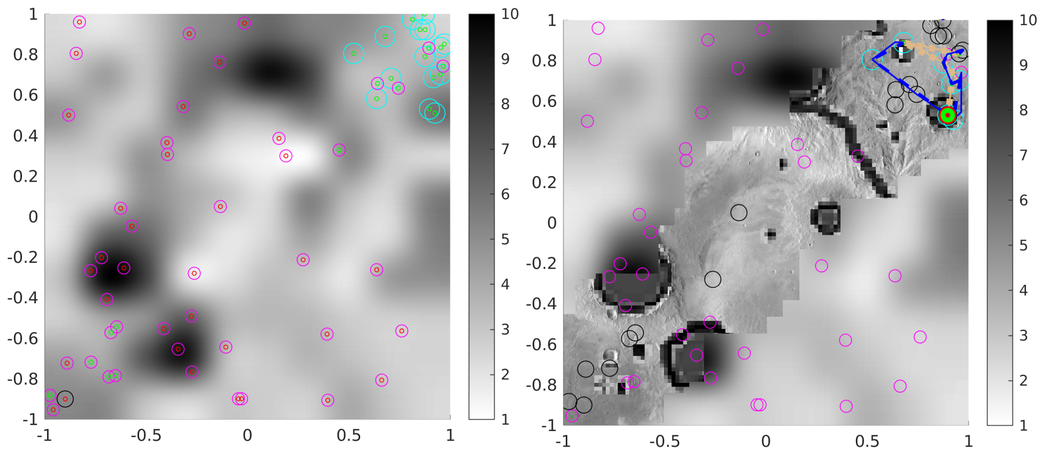


Figure 3.2: Starting state and a potential ending state for a planetary orienteering problem with a map revealing. The black-to-white gradient (the darkest areas are hardest to traverse) encodes the cost to traverse. This figure demonstrates the change of local information gained during operation compared to the initial noisy prior map.

3.5.1 Simulation setup

The simulation is motivated by the problem of a rover exploring a planetary terrain with limited surface-level information, such as mapping and sampling the surface of Europa or Enceladus.

Mapping and sampling these surfaces are of high scientific interest due to the potential to contain extraterrestrial life or provide additional information about the formation of our planet [212].

However, these types of problems are difficult for a multitude of reasons; in this work, we focus on the main constraints of limited available map information, limited communication availability, and weight constraints. Map information is limited and noisy due to the availability of orbital satellite imagery that is sufficiently detailed for surface-level operation. Limited communication restricts the effectiveness of teleoperation or centralised algorithms, so proposed systems either need to wait for communications or operate decentralised. Lastly, space mission weight constraints limit the amount of computational power available by restricting the physical mass of computing components and the required energy to operate the processing. Since our scenario is not motivated by any specific instance of the planetary exploration problem, we generate a synthetic demonstrative scenario to investigate the applicability and performance of our proposed method in a problem scenario similar to the conditions we focus on in this work.

For this problem, agents need to traverse a graph representation of the environment $G = (V, E)$ with a limited budget B and visit as many POIs as possible Q. The costs of traversing edges are given by a noisy map, which is converted into a costmap C that contains the costs for traversing the edges in E. As a robot traverses the environment, it samples its local area and updates the costmap C to reflect the ground truth costmap C^*.

To simulate this environment, we create a synthetic low-resolution noisy satellite map from a higher-resolution image. Our simulated map began with a false colour high-resolution map of Mars orbital data [213], which was then modified with the process shown in Figure 3.3. The original image was converted into a costmap by converting it to greyscale and then thresholding it; the resulting grey levels of the new image became the costmap. This cost map was used to compute the ground truth costs C^*. This ground truth map was then down-sampled and Gaussian blurred to create the low-resolution noisy orbital map. This noisy map is the initial understanding of the environment the system uses to plan, and using local observations, this map is updated.

Experiments were performed with one simulated physical robot and a ground station that planned for ten virtual agents. The α intuition weighting parameter was set to 1, which balances the reward between following the guidance and the robot following its own plan. The rollout weighting

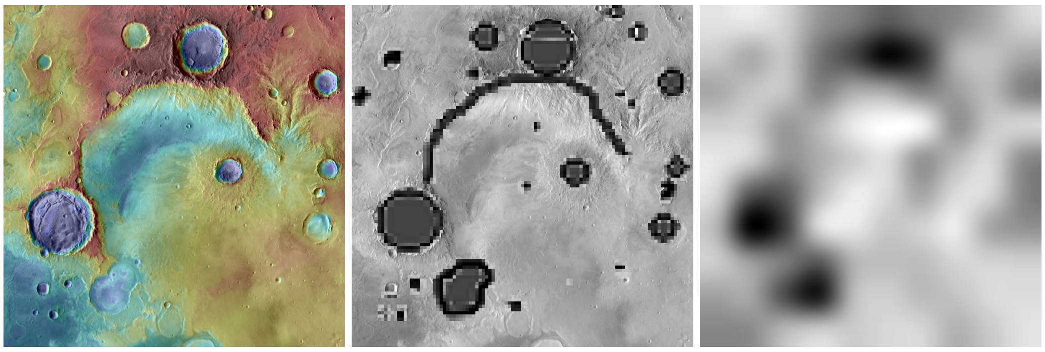


Figure 3.3: The process of creating the synthetic planetary cost map. (Left) Original false colour Mars orbital map. (Middle) Map after being greyscaled and thresholded to convert it to a cost map. This map is used as the ground truth for the planner. (Right) Map after being down-sampled and Gaussian blurred to create a synthetic low-resolution orbital map that is used as the prior for the planner.

parameter λ is set to 0.5 to balance the probability's distance and pheromone guidance components. The MCTS was run for 250 iterations for the physical robot and for 10000 iterations for the ground station to model their difference in computational power. Each random problem instance contained 50 Points of interest V with rewards of either 1 or 3 points of utility Q. A PRM was built using 1000 vertices with the cost of moving between points c_{ij} computed using the current map state. Upon receiving new information, the costmap state was updated for the agent. Each problem instance is a 5-day long mission with communication between the ground station and mobile robot happening once at the start of each day. The once a day communications are similar to the communication restrictions imposed on a proposed mission concept for a Europa lander mission [214]. The ground station planning horizon was the entire length of the mission, while the mobile robot focused on the day ahead. Subsequently, the mobile robot has a $B = 1$ while the virtual agents planned using a budget for the length of the remaining mission $B' = (5 - d) * B$, $B = 1$, where d be the days remaining in the mission. If the physical agents ran out of budget while executing their plan, the agent was stopped where its budget ran out, and the next day's plan continued from this point.

We compare against two algorithms that help demonstrate our methods' benefit. First, we compare against the physical robot using its limited computing to plan its daily operations without the influence of the ground station; we label this approach as *Solo*. Next, we compare against planning a path for the robot from the ground station without considering the robot's decision-making and executing it blindly. This plan is computed over the whole mission's length without consideration for the new information that will be gained. This second approach is labelled as the *Long horizon*.

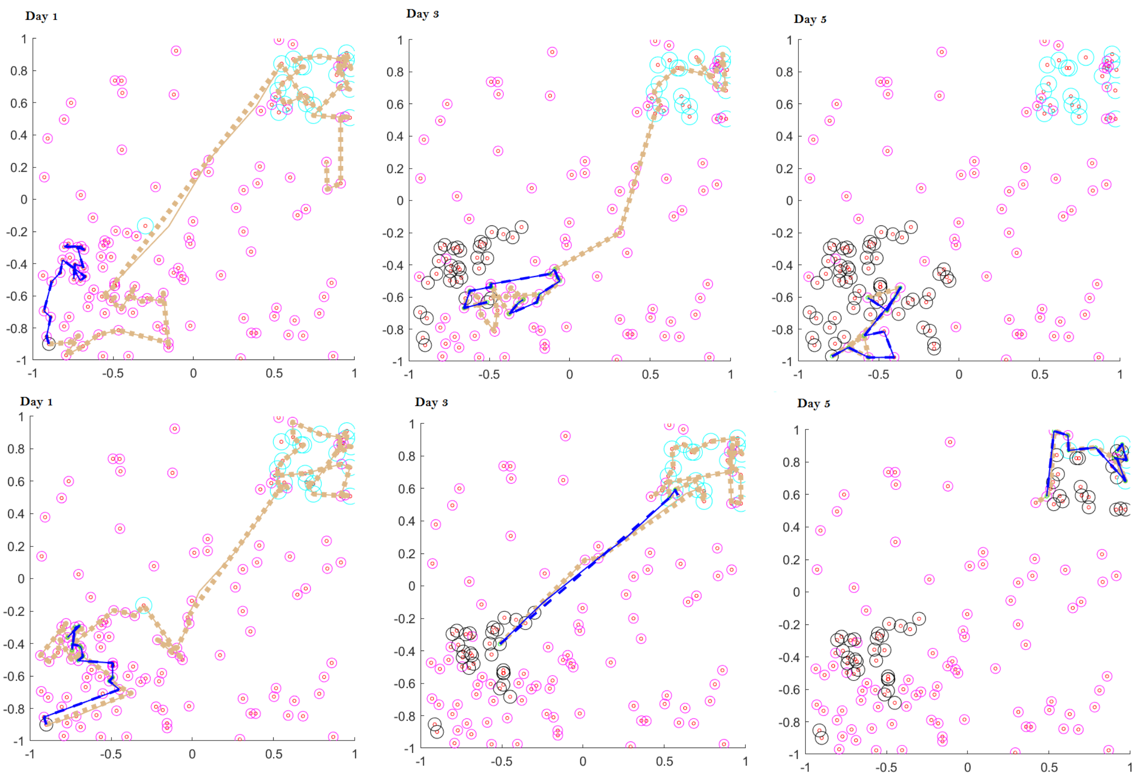


Figure 3.4: Five-day long orienteering mission for one physical and one virtual guiding agent. The (top row) shows the mobile robot planning one day at a time without influence from the ground station. The (bottom row) shows our joint method, where the virtual agents influence the mobile robots' choices. It can be seen from these illustrative examples that our method is capable of leveraging intuition to make less myopic decisions despite both methods having the same level of computation for the mobile robot. Pink circles are one worth one point of reward, cyan circles are worth three points, and black circles are already visited. The blue line is the physical agent, and the orange is the virtual guiding agent(plans with five days of budget)

3.5.2 Results

The collated results for 100 random problem instances (Figure 3.5) were generated using our simulation parameters. These results show that our method is able to achieve a higher number of visited points than either of the two comparison methods. This demonstrates that combining sequential local planning with guidance from the long-horizon plans was beneficial. Our method improved the number of visited points by 10% over the long horizon plan and by an average of 30% over the myopic solo sequential greedy planner.

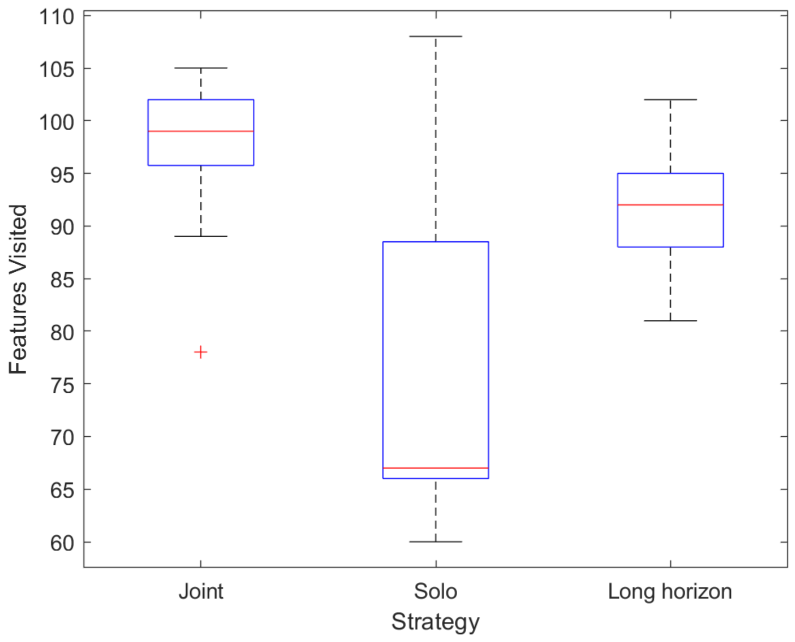


Figure 3.5: Collated results for 100 random problem instances comparing the number of gathered rewards over a 5-day long mission. It can be seen from these results that the robot acting on its own in the 'Solo' strategy has a high variance, while the less informed 'long horizon' strategy is similarly limited by its lack of new information. Our joint method that uses the long horizon plans to inform the local decision-making has better performance characteristics.

The 'Solo strategy' method does not have the influence of intuition, and thus, its variance is much higher due to the planner only focusing on optimising its short horizon. This effect is shown in Figure 3.4 the top row demonstrates that when using our method, the behaviour of the ground robot was guided towards taking a less optimal choice for a time, which resulted in an overall better result for the mission. The second row shows the myopia of the locally greedy method and that it is not able to see far enough into the future (due to its limited computation) to be able to reach the area of higher rewards.

Similarly, the 'long horizon' results show that our method can perform better than if a more detailed plan were forced onto the ground robot without considering the benefits of local information. The local robot is better able to react to the new map information and better estimate the costs of traversal, which leads to being able to find better local paths.

3.6 Summary

We have presented a solution to the Discoverable Edge Cost Orienteering problem (DECOP), which is a type of problem with a stochastic objective. To solve this problem, we perform replanning using information gathered online, and to overcome the myopia of replanning, we want to carefully leverage the additional computation of a ground station to inform the mobile robots' decisions. The mobile robots' decisions are influenced by the pheromones that represent virtual agents controlled by the ground station. These pheromones act as an intuition that biases the mobile robots' search towards good solutions that have been planned over a longer horizon. This method is based on a modified Dec-MCTS algorithm that finds solutions for the team of mobile robot and virtual agents. Finally, we demonstrated our method's effectiveness in a planetary exploration where the mobile agent has limited computing and the environmental understanding is initially noisy. Through simulation, we show that our method performs better than solving the DECOP problem with the limited computing of the mobile robot or the extensive computing but limited information available to the ground station.

This chapter investigated stochastic objectives through the process of solving DECOP. In this problem, the uncertainty in the objective function was caused by the limited budget and the noisy map of the environment. Given the unknown cost of traversing the environment and the limited budget, it is unclear how much of the planned sequence of observations is possible. Subsequently, the reward for a plan is stochastic. This problem could be avoided with perfect map information or perfect map prediction. However, it is impossible to create a perfect prediction of the real-world map, so instead, the uncertainty in the objective is reduced by incorporating new information as it becomes available. Through replanning, we adapted the plans to include the local information gathered by the mobile agent during the operation. Although replanning can be myopic, the collaboration of the external computation helped to overcome this limitation.

In the next chapter, we investigate scheduling problems where stochastic changes in the environment or system modify the constraints. Due to the stochastic constraints, valid plans can become invalid when the constraints change. This is in contrast to stochastic objective problems, where the uncertainty results in less reward for a given plan. In stochastic constraint problems, the tasks are fully known, and the reward for taking actions is fully known, but a plan can become invalid due to the constraint changes. The key idea is to try and find a way to make plans feasible despite the changing requirements.

Chapter 4

Stochastic Constraints

This chapter covers the second component of our probabilistic scheduling framework scheduling operations for systems with stochastic constraints. Constraints define the rules of the environment and determine if a set of decisions is valid. Stochastic constraints occur in scenarios where the rules of a system change and are driven by external stochastic processes. The tasks that need to be completed are fully known, the objective function is deterministic, and no stochastic consumable resources need to be considered. However, due to the constraints, changing a schedule may be valid in one instance of the constraints, but under a different configuration, the same schedule may be invalid. This invalidation of previously feasible plans is more restrictive than the stochastic objective case, as the uncertainty causes failure (which returns no reward) rather than suboptimality.

This chapter introduces this class of problems in an oceanographic information-gathering scenario to explore the idea of stochastic constraints. In this problem, we need to find a joint schedule for a team of vessels operating in the ocean that covers all points of interest using a novel Multi-Vessel Multi-Float (MVMF). This problem is a type of stochastic constraint scheduling problem. In this variation, the team needs to minimise the makespan of visiting all the points of interest. To observe the points, the vessels deploy underactuated floats carried by the ocean currents, which need to be collected during a window in the future when they surface. The chaotic nature of ocean currents makes the timing windows of valid schedules stochastic. We develop a hierarchical solution to this problem by decomposing it into two smaller problems. The solution to the first gives the smallest valid set of needed deployments, and then the pickup and dropoff sequence of operations is computed for the team. This method was tested through simulation, demonstrating its suitability to the problem. Subsequently, it was tested with real hardware in the field to validate the system and observe its limitations during practical field operations. Finally, we briefly mention the preliminary results for an extended field trial with replanning added to our algorithm to account for a limitation observed in the original field trials.

4.1 Overview

The focus of this chapter is dealing with problems where the constraints that define the system are stochastic. This difficult-to-address formulation occurs in practice when the changes in the environment or system cause the solution to become invalid rather than reduce the utility. We demonstrate this class of problem in an oceanographic information-gathering scenario. Due to the realities of operating robots in the ocean, the currents can cause planned operations to become too risky and thus invalid.

In this chapter, we formalise the notion of coordination between actuated vessels and low-cost, energy-efficient floats. The *Multi-vessel Multi-Float (MVMF)* problem is finding valid drop-off and pick-up locations so deployed floats can visit given points of interest at known locations. Its central challenge is finding a sequence of trajectories over a continuous oceanic flow field that allows the floats to observe the points of interest. MVMF can be viewed as a type of pickup and delivery problem vehicle routing problem (VRPPDP), known as NP-hard [182]. Additionally, to preserve the safety of the platforms, a tardiness constraint, i.e how long the float is allowed to be left unattended at the surface, must be enforced. This constraint means that valid sequences of operations are a function of the ocean currents. Oceanic currents are highly chaotic and complex to model, so the constraints are effectively stochastic.

Autonomous underwater vehicles (AUVs) play a critical role in many marine science applications such as underwater habitat mapping [215–217], environmental monitoring [218–221], geological surveying [222–224], and plume source detection [225–227]. An exciting opportunity to increase the accessibility and availability of marine autonomous systems arises from the idea of replacing high-cost, fully featured AUVs with a team of under-actuated robotic *floats* [228–230] as shown in Figure 4.1, which can control their depth but otherwise drift according to the surrounding ocean

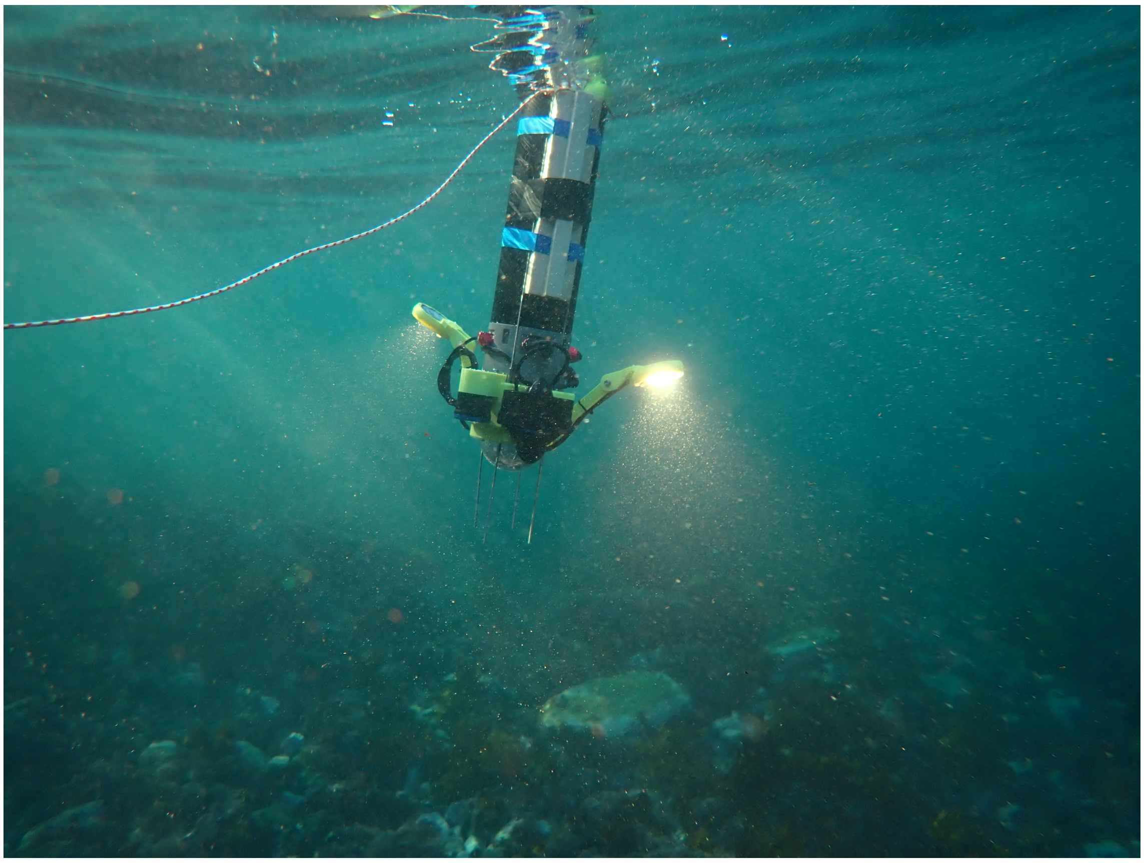


Figure 4.1: Ocean float taking images of underwater features in a field trial for Schmidt Ocean Institute (Clovelly, NSW, Sydney).

current. Multiple floats can work in collaboration with a small group of surface vessels, which deploy the floats at locations such that they drift into a given region of interest to collect data. Then, the floats await pick-up by the surface vessel to be serviced and redeployed. In general, each surface vessel can support the operation of multiple floats. This approach moves most of the complex instrumentation and decision-making to the surface, where it can be readily supervised by humans while reducing the cost of the equipment at risk underwater and offering some redundancy to loss by operating multiple underwater assets.

To solve the MVMF problem, we present a hierarchical approach that provides a practical solution. Our approach first uses the Monte Carlo tree search (MCTS) algorithm to find a set of float trajectories that maximise the number of *point of interest* (POI) collectively observed. Then, we again use Decentralised Monte Carlo Tree Search [111] (Dec-MCTS) based algorithm to find each surface vessel's drop-off and pick-up schedule. We assume that the floats are spatially dispersed, so the chance of collision is low and can be ignored. Further, given that these floats are designed to be small and relatively inexpensive, they do not carry much energy, so we assume that mission time is sufficiently brief to consider the oceanic flow field as time-invariant for the duration of the mission. The technical novelty of our approach is to introduce necessary computational specialisations to the general Dec-MCTS framework for sampling, rollouts, and action generation. The size of the state space is a function of the number of points of interest, floats, vessels, and candidate trajectories and is too large for the naive application of existing algorithms.

We formally define the MVMF problem and present the details of our hierarchical algorithm, along with an analysis of its optimally, correctness and completeness properties. To evaluate the algorithm's performance, we report simulation experiment results showing that our solution is computationally efficient for practical problem sizes, is flexible enough to admit mission constraints, and produces solutions with acceptable solution quality. These results pave the way towards conducting initial experiments with this novel and promising class of marine robot systems.

Following the verification of the simulations, our hierarchical planner was tested in the field on a MVMF system. These field trials were useful to validate the system. The insights gained from the field trial were helpful in understanding the directions that our method needs to be improved in to improve its results during field operations. The main insights gained were around the suitability of assuming the ocean is temporally invariant over short time horizons; this motivates replanning to adapt to changing conditions. Additionally, it motivated the further analysis of 3D flow field estimation due to the differences between 2D and 3D incompressibility of water assumptions need to be carefully considered.

Lastly, a simple replanning extension was added to the operation of our MVMF algorithm, and preliminary further field trials were performed to see if the insights gained from the first set of experiments were sufficient.

4.1.1 Chapter outline

The remainder of this chapter is organised as follows. Section 4.2 defines the Multi-Vessel Multi-float problem and compares it to a traditional scheduling problem and typical VRP formulations, showing why this new form is necessary. Section 4.3 describes our hierarchical-based algorithm for solving the MVMF problem in a scalable way. Section 4.4 validates our system in simulation

Section 4.5 describes the hardware field trial that we performed in Jervis Bay with a MVMF system. It also outlines some of the lessons we learned from these trials. In Section 4.6 we describe the online replanning extension and the preliminary field trial that included replanning. Finally, Section 4.7 summarises the chapter.

4.2 Problem formulation

In this section, we present the problem formulation for the MVMF scheduling problem for a team of information-gathering vessels operating in the ocean. In this problem, the goal is to visit all the Points of Interest (POIs) while subject to temporal constraints caused by chaotic ocean flow fields. We first model the problem as a generic vehicle routing problem with time windows and show that this formulation is insufficient to capture the specific complexity of this problem and is too hard to solve. We then formally define a specific scheduling problem form for the MVMF problem that is more feasible to solve.

4.2.1 Robots and envrionment

Let $f_c : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ be a time-invariant fully-known oceanic flow field in A 2D environment. We have a discrete set of *points-of-interest* (POIs) $\mathbf{Q} \subset \mathbb{R}^2$ in the environment. The POIs are static and fully known in advance. We are interested in maximising the number of POIs visited using two types of ocean vehicles: *surface vessel* $\alpha \in \mathbf{A}$ and *float* $\beta \in \mathbf{B}$.

In the environment, we have a set of *surface vessels* $\mathbf{A}$ and *floats* $\mathbf{B}$. We assume that surface vessel $\alpha \in \mathbf{A}$ is fast enough to ignore flow vectors, such that it moves from one position to another in a straight line with ground speed v_α, such that

$$\dot{\alpha} = v_\alpha \cdot \begin{bmatrix} \cos\theta_\alpha \\ \sin\theta_\alpha \end{bmatrix}, \tag{4.1}$$

where θ_α is travel direction. On the other hand, a float $\beta \in \mathbf{B}$ has no actuation so that its dynamic is solely influenced by flow field f_c, such that

$$\dot{\beta} = f_c(\beta). \tag{4.2}$$

For further details on the system dynamics and flow field environment, see Appendix A.

Surface vessel $A_i \in \mathbf{A}$ has a number of floats assigned such that $B_i = \{\beta \in \mathbf{B}\}$ is a set of floats assigned to vessel α_i where no float is assigned to multiple surface vessels at the same time (i.e., $B_i \bigcap B_j \equiv \emptyset$ for all $A_i, A_j \in \mathbf{A}$ and $i \neq j$). A set of *deploy actions* for surface vessel α_i is denoted as $\mathcal{A}_i \in \{D_{i,k}, P_{i,k}\}$ where $D_{i,k}$ and $P_{i,k}$ denotes *drop-off* and *pick up* for assigned float $\beta_k \in B_i$. We slightly abuse notation for $D_{i,k}$ and $P_{i,k}$ to include the position where the action is taken.

4.2.2 Generic vehicle routing with time windows form

Modelling the MVMF problem as a classic machine shop scheduling problem is difficult due to the interdependence of the costs of servicing tasks being a function of the previously serviced task. As the vessels move around the environment, the cost to visit points depends on where they are. The problem would need to capture the sequence-dependent job starting costs and the variable time windows introduced by the pickups. Alternatively, the problem could be modelled as a vehicle routing problem. The Vehicle Routing Problem with Time Windows (VRPTW) is a suitable variant. If we consider the surface vessels as vehicles, they are moving to service jobs (drop-offs) that cause an associated pickup location with a time window to occur. This new pickup job must be completed during the time window, or the schedule is invalid. Modelling this as a VRP is difficult due to VRPs changing starting location constraints, and the problem is still NP-Hard to solve. Thus, we find it more useful to address instead where the problem complexity is coming from and formulate a new specific problem for this class of problem.

Even though the set of tasks is fully known, one of the main difficulties comes from the fact that the service time of the subsequent pickup task is unknown because the float can resurface (functionally) anywhere due to the chaos of the flow field. To solve this problem, we would need to know the flow field perfectly and its evolution over time. With this knowledge, it would be possible to disambiguate the pickup locations and thus resolve the time windows for the pickup jobs, which allows for avoiding the stochastic constraints. To do this, we assume that the timescale for changing the flow field is large enough that when considered over a short horizon (approximately hour), it can be assumed to be quasi-static. If we observe the flow field at the current point in time and then lock it in place for a short window, we can use this to compute a schedule. This is a form of a single step-ahead prediction conditioned on the current state of the flow field. That is, we assume the short-term future state is the same as the now.

4.2.3 MVMF scheduling form

We define *schedule* $\boldsymbol{\phi}_i$ as a sequence of deploy actions $\mathcal{A}_i$ for surface vessel α_i. For example, $\boldsymbol{\phi}_i = D_{i,1}D_{i,2}P_{i,2}P_{i,1}$ implies that float 1 and 2 are dropped off in sequence at their designated positions and float 2 is picked up before float 1. If the next deploy action is to pick up a float, then the surface vessel moves to pickup position $P_{i,k}$ and waits until the float arrives. Time waiting for float k is denoted as wait time $\tau_{i,k}$. A schedule is said to be *valid* iff every wait time $\tau_{i,k}$ in the schedule is equal to or greater than zero; otherwise, the vessel is tardy, and the pickup has failed (and possibly leading to the float being lost). Also, the pick-up action $P_{i,k}$ cannot be done before $D_{i,k}$. The corresponding controls (i.e., travel directions) for surface vessels can be easily derived from (Equation 4.1). The set of POIs visited by schedule $\boldsymbol{\phi}$ is denoted as $Q(\boldsymbol{\phi})$.

We are interested in a problem where we find a joint schedule $\boldsymbol{\Phi} = \{\boldsymbol{\phi}_\alpha\}_{\alpha\in\mathbf{A}}$ for a team of vessels $\mathbf{A}$ so that all unique POIs are visited by a set of floats $\mathbf{B}$ in minimal makespan time. Let $T(\boldsymbol{\Phi})$ denote the makespan incurred by the joint schedule. We formally define the problem as follows.

Problem 4.1 (Multi-Vessel Multi-Float Scheduler)**.** *Given a time-invariant oceanic flow field f_c, set of surface vessels $\mathbf{A}$ and floats $\mathbf{B}$, find a joint schedule $\boldsymbol{\Phi}^*$ that it visits all POIs with minimal makespan time:*

$$\begin{aligned}
&\boldsymbol{\Phi}^* = \arg\min_{\boldsymbol{\Phi}} T(\boldsymbol{\Phi}) \\
&\textit{s.t. } |\bigcup_{\alpha\in\mathbf{A}} Q(\boldsymbol{\phi}_\alpha)| = \mathbf{Q} \\
&\textit{and } Q(\boldsymbol{\phi}_{\alpha_i}) \cap Q(\boldsymbol{\phi}_{\alpha_j}) = \emptyset, \forall \alpha_i, \alpha_j \in \mathbf{A}.
\end{aligned} \tag{4.3}$$

4.3 Hierarchical schedule framework

In this section, we present the hierarchical framework that solves the surface vessel schedule that maximises coverage of POIs with minimal makespan. We first reduce the size of the search space by sampling a set of drop-off/pick-up actions over the 2D time-invariant flow field f_c with a discrete set of POIs $\mathbf{Q}$. Subsequently, we use the subset of sampled actions and a Dec-MCTS-based algorithm to find the decentralised single vessel schedules that together comprise the joint schedule. This joint schedule minimises the overall mission makespan for the set of surface vessels $\mathbf{A}$ and floats $\mathbf{B}$. Finally, we present a brief analysis on the validity, scalability, and optimality of our method to solve the desired problem.

Even though the stochastic constraints have been dealt with in the problem formulation, the search space that needs to be explored to find a valid schedule is still massive. The search space also grows exponentially with the number of floats and surface vessels. As such, we address the problem using a hierarchical method by splitting the main problem into two sub-problems and then solving them sequentially:

1. Given a time-invariant oceanic flow field f_c, set of POIs $\mathbf{Q}$, set of sampled drop-off actions $\mathbf{D}$ and its corresponding set of pick-up actions $\mathbf{P}$, find a set of drop-off actions $\mathbf{D}_s \subseteq \mathbf{D}$ and its corresponding pick-up actions $\mathbf{P}_s \subseteq \mathbf{P}$ that observes all unique POIs in $\mathbf{Q}$.
2. Given a set of surface vessels $\mathbf{A}$ and floats $\mathbf{B}$, and the sets $\mathbf{D}_s$ and $\mathbf{P}_s$ found in the previous sub-problem, find the joint schedule $\mathbf{\Phi}$ over all $\mathbf{D}_s$ and $\mathbf{P}_s$ that minimises the makespan of the overall mission.

We make the following assumptions throughout the method. First, we assume that the floats and surface vessels are spatially dispersed during operations, and therefore ignore collisions between vessels and floats. Local planners can also be used to avoid collision cases. We also assume that, while the float operates in a 3D oceanic environment, the problem is projected as a 2D problem by taking the depth-averaged flow field. This assumption is common in existing work [231–233].

We solve the two sub-problems using methods that are based on *Monte Carlo tree search* (MCTS) [234]. The MCTS algorithm is a sampling-based biased search algorithm that is suitable for large state space problems with non-trivial objective functions, such as in the MFMV problem. MCTS begins from an initial condition and then iteratively grows the search tree by selecting the next state to explore from a set of actions to take from the selected state. From the selected state, a sequence of actions is randomly sampled using a heuristic (e.g., biased random sampling) until the termination condition has been met (e.g., length of sequence); this process is called a *rollout*. The score from the rollout is then used to update the tree's expectations, and the process continues.

The decentralised extension to MCTS, namely *decentralised MCTS* [111], is a modification to the MCTS algorithm that allows multiple agents to coordinate to solve a joint objective function for the whole team. Each agent finds its own solution by growing the MCTS tree while considering the set of potential actions by other agents. The algorithm allows for finding an asymptotically optimal joint set of solutions for the whole team.

In the proposed hierarchical method, the first sub-problem of selecting the set of pick-up and drop-off actions from a large number of samples is done using MCTS. For the second sub-problem, the subset of actions is then scheduled and allocated to the surface vessels using Dec-MCTS, where each surface vessel independently computes its local plan, which leads to the joint optimal solution.

4.3.1 Float trajectory selection

The selection of possible drop-off/pick-up actions is done in two steps: sample candidate drop-off actions, then select a subset of actions that observe all the features in the smallest number of actions. The sampling step samples N number of drop-off locations $d \in \mathbf{D}$ across the oceanic environment f_c populated with POIs $\mathbf{Q}$. For each drop-off location, the corresponding pick-up location $p \in \mathbf{P}$ is derived from (Equation 4.2) over some time-bound Δt, and function $\mathcal{Q}(d, \Delta t)$ determines the subset of POIs observed $\mathbf{q} \subseteq \mathbf{Q}$ from this pair of deploy actions.

The selection step uses an MCTS algorithm to choose a subset from the sampled drop-off actions $\mathbf{D}_s \subseteq \mathbf{D}$ and the corresponding pick-up actions $\mathbf{P}_s \subseteq \mathbf{P}$ that observe all POI in the minimal number of actions. To speed up the convergence rate in the rollout process, we use a greedy sampling heuristic that selects trajectories that cover unobserved POIs more often than other trajectories. For this, when we select a dropoff action, we select its corresponding pickup action Let $\mathbf{D}'_S$ be the currently selected subset of actions during the rollout. The likelihood of selecting action d_s during the rollout can be computed using:

$$P(d_s|\mathbf{D}'_s) = |\mathcal{Q}(d'_s) \cup \mathcal{Q}(\mathbf{D}'_s)| + \epsilon \tag{4.4}$$

where we omit the Δt from the $\mathcal{Q}$ function for clarity.

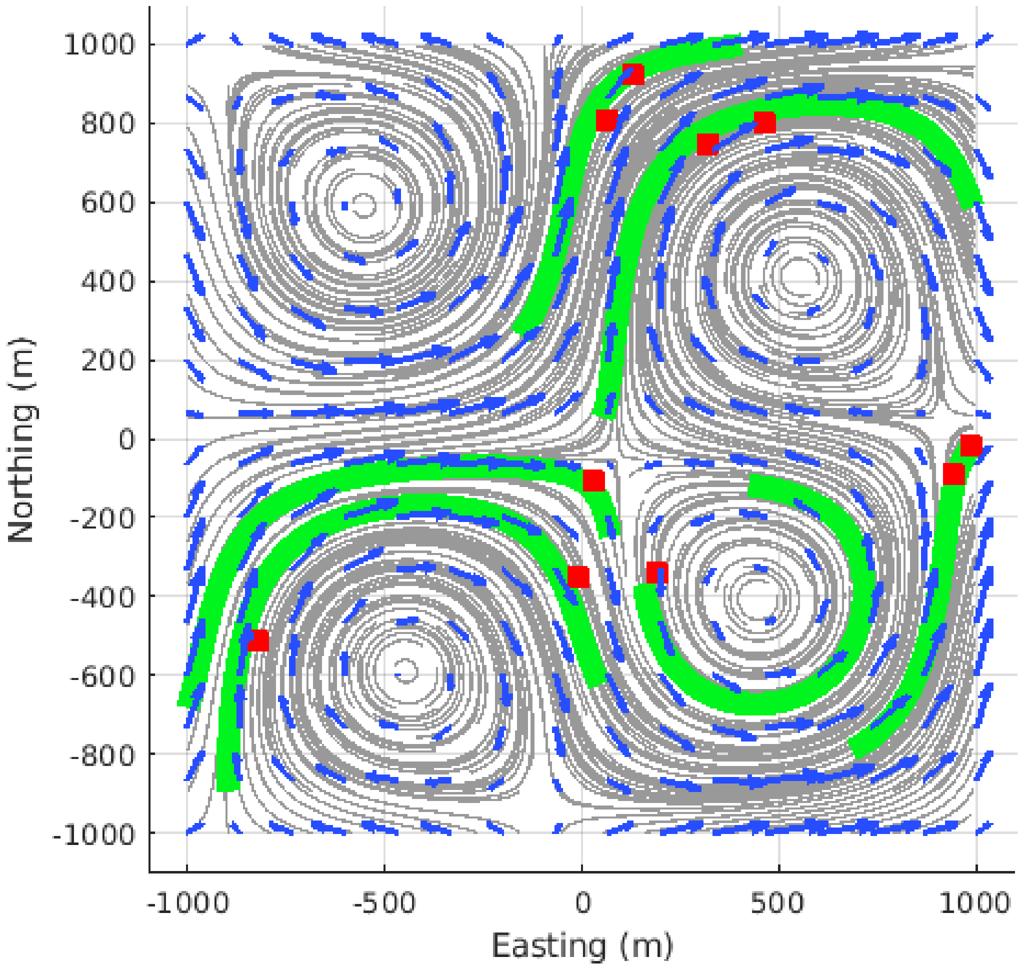


Figure 4.2: From the large set of sampled trajectories (shown in grey) generated from the oceanic environment (shown in blue quiver), a minimal covering subset has been selected (shown in green) that visits all POIs (shown as a red marker).

The utility of a drop-off location d over time-bound Δt is defined by the number of observed POIs over the total number of POI $|\mathbf{Q}|$. The reward function for the MCTS is the total utility of the set of drop-off actions minus a penalty for the number of drop-off actions taken. This drives the algorithm towards observing POIs that are not yet observed while using the fewest number of possible drop-off actions. An example is shown in Figure 4.2.

4.3.2 Surface vessel scheduler

In this section, we address the second sub-problem of our hierarchical approach. We present a surface vessel scheduler that produces schedules of drop-off and pick-up actions for the surface vessels and their floats.

Given the set of deploy actions that observes all POIs, the set of surface vessels and floats, we wish to find the joint schedule $\mathbf{\Phi}$ of float drop-off and pick-up actions that minimises the makespan of the mission. Finding optimal solutions for a coordinated team of vessels is difficult because it requires planning over all possible combinations of the vessels' actions, which is known as planning in the *joint space*. Decentralising allows each vessel to plan its schedule in the much smaller single-vessel search space, hence faster search for good solutions. We compute surface vessel schedules that comprise the joint schedule using a Dec-MCTS-based algorithm. To help with convergence, we bias the rollout function based on the proportional distance to each action from the current position of the vessel in its schedule. Additionally, the rollout is biased, but not restricted, to select actions that are not yet selected by other vessels to encourage the team to select actions the team has not yet done. Formally, given a partial single vessel schedule $\boldsymbol{\phi}'$ and the set of actions that are not in this schedule $\mathcal{A}^{(\prime)}$. Let $a^{(\prime)}$ be a single action $a^{(\prime)} \in \mathcal{A}^{(\prime)}$, let g' be the position of the vessel during $\boldsymbol{\phi}'$ which is equivalent to the position where the last action will be taken. Let other actions that are allocated to the other vessels be $\mathcal{A}^{o}$ which can be estimated from the Dec-MCTS messages. Let μ be a binary variable that represents if an action is taken by another agent in the joint schedule, $\mu = 1$ if $a^{(\prime)} \in \mathcal{A}^{o}$ else $\mu = 0$. The rollout biasing is computed using:

$$p(a^{(\prime)}|\boldsymbol{\phi}') = L_2(g', a^{(\prime)})\lambda_d + \mu\lambda_b + \epsilon. \tag{4.5}$$

Where L_2 is the L2 norm and λ_d controls the weighting for selecting actions that are nearest to g' while λ_b influences the vessel taking new actions that are not in the joint schedule behaviour.

The utility function for the scheduler rewards the allocation of actions and penalises makespan for the overall mission. The utility function also sets a hard constraint regarding the vessel pick-up tardiness of the joint schedule $\mathcal{W}(\mathbf{\Phi})$. This enforces that $\tau_{i,k} > 0$ for all vessels in the schedule $\mathbf{\Phi}$. Given the number of actions taken from the subsets $|\mathbf{D}_s| + |\mathbf{P}_s|$, a makespan time weighting parameter λ_M, and makespan of the schedule Δ_M. The utility function with respect to the number of actions taken S and the total makespan is defined as:

$$\begin{cases} 0 & \textbf{if } \mathcal{W}(\mathbf{\Phi}) > 0 \\ \frac{S}{|\mathbf{D}_s|+|\mathbf{P}_s|} + 1 - 2^{\lambda_M \Delta_M} & \textbf{otherwise} \end{cases}. \tag{4.6}$$

For a joint schedule, if one vessel is tardy, it invalidates the entire joint schedule. Naively allowing any vessel to consider any drop-off action can result in a significant number of schedules that contain pick-up actions that exceed the tardiness bound. While the planner can technically find valid schedules by generating random sequences of pickups and drop-offs, the search would be inherently slow as it also naively considers many invalid actions. Even though all drop-off actions may be valid, they may not be feasible with respect to the overall schedule because that drop-off action may delay the vessel from picking up the float from its previous drop-off action, resulting in tardiness. To speed up convergence, we introduce a modification to the MCTS action generation that allows vessels to only plan schedules that contain drop off actions that are feasible without invalidating the tardiness bound.

4.3.3 Action generation

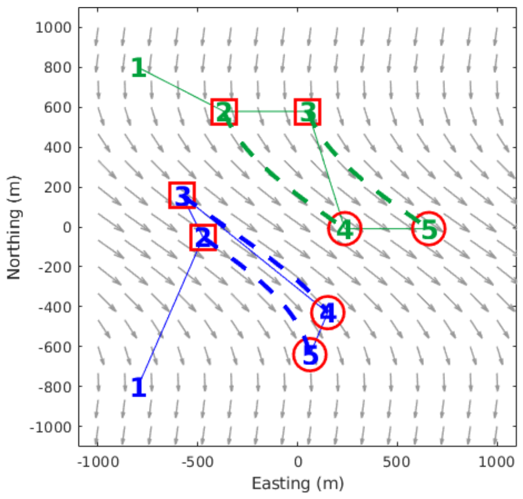
If all floats for the vessel are already deployed and not yet picked-up in $\boldsymbol{\phi}'$ no more drop-off actions can be added, otherwise to add a new surface vessel drop-off action $d^{(\prime)} \in \mathbf{D}_s \cap \boldsymbol{\phi}'$ we use the following to check if it is feasible. Let $\mathbf{P}^{(\prime)}$ be the set of to be completed pickup actions based on the matching the drop off actions $\mathbf{D}' \in \boldsymbol{\phi}'$ and $p^{(\prime)} \in \mathbf{P}^{(\prime)}$ is the pickup action in the pickup action set. Let $\mathcal{T}(\boldsymbol{\phi}', p^{(\prime)})$ be a function that returns the time remaining before $p^{(\prime)}$ becomes tardy based on the partial schedule. Find the tightest bounding time:

$$TB = \underset{p^{(\prime)} \in \mathbf{P}^{(\prime)}}{\arg\min}(\mathcal{T}(\boldsymbol{\phi}', p^{(\prime)})). \tag{4.7}$$

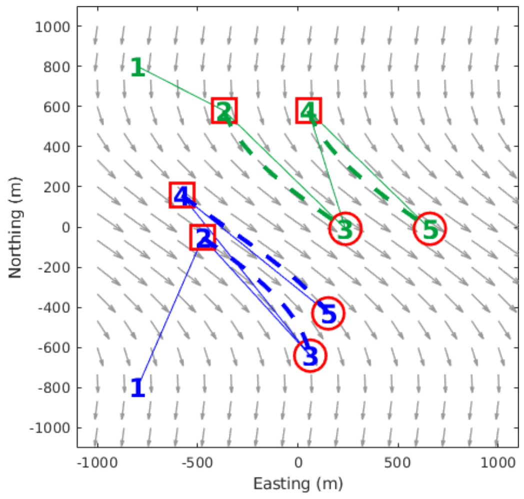
With this bound the validity of adding $d^{(\prime)}$ to the schedule can be checked using:

$$\frac{L_2(g', d^{(\prime)} + L_2(d^{(\prime)}, p^{(\prime)}_{TB})}{v_a} < TB. \tag{4.8}$$

Where $p^{(\prime)}_{TB}$ is the pickup with the tightest bound. These equations ensure that drop actions will be added to the partial schedule and will not invalidate the tardiness bound. Additionally, it decreases the number of candidate actions the planner needs to consider when producing schedules which speeds up convergence and allows our algorithm to produce quality solutions faster.



(a) Hierarchical schedule results for 2 vessel 2 floats



(b) Sequential greedy result for 2 vessel 2 float

Figure 4.3: (a) shows the sequence of drop off and pick up actions for a two vessels carrying two floats using Dec-MCTS. (b) shows the sequence of drop off and pick up actions for two vessels carrying two floats using sequential greedy. Sequential greedy can only utilise a single float. The surface vessel begins at 1 and moves to subsequent actions to complete the drop off(square) and pickup (circle) actions.

4.3.4 Analysis

The problem of finding optimal joint schedule $\mathbf{\Phi}^*$ is computationally hard due to its inherently large search space. Naively, the action space is given over a continuous flow field f_c where the drop position can be anywhere but the pickup position is constrained by flow field $P_{i,k} = \dot{\beta}(D_{i,k})\Delta t$. Since finding float trajectory in the presence of a flow field is computationally hard (i.e., no analytical solution is known) and there exists an infinite set of possible drop positions, a naive approach is intractable. Instead, the proposed framework reduces the search space to a finite set of discrete actions that are in the form of float trajectories. Since the performance (i.e., the number of POIs visited) for each float trajectory is pre-computed, the search for an optimal joint schedule is computationally scalable in the size of the environment and the number of floats. Furthermore, since the scheduling step is decentralised across surface vessels, the overall computation also becomes scalable in the number of surface vessels.

We guarantee that any given schedule $\mathbf{\Phi}$ is *valid* by ensuring that if a schedule is not valid (i.e., sequences of actions cause a vessel to be tardy $\tau_{i,k} < 0$ when picking up a float), it has 0 utility; thus the scheduler will not consider it a valid solution. Similarly, the valid action constraint in (4.8) does not allow adding drop actions that invalidate the current considered partial schedule. Since we ensure that only a valid schedule is generated at each iteration, we preserve an important algorithmic property in MCTS called *anytime* property. The property implies that scheduling can be stopped at any time to get a valid solution. More computation time can be allowed to find a better solution.

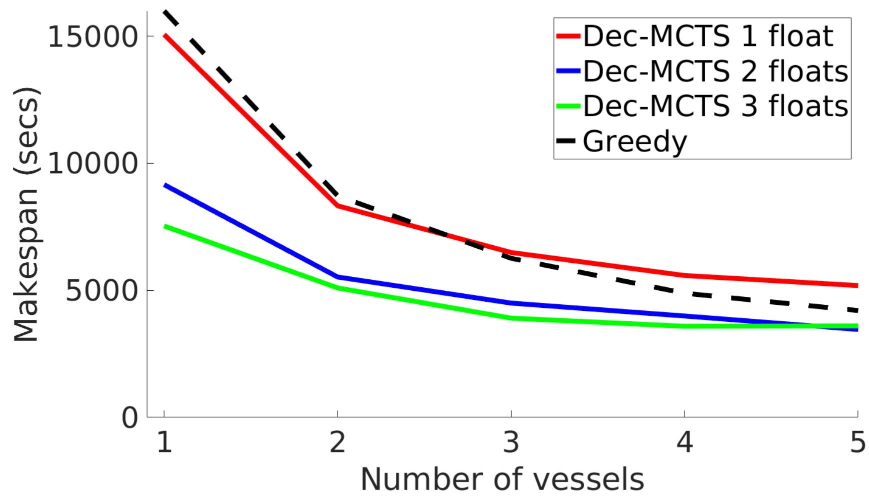
For our solution, each step in the hierarchy is asymptotically optimal with respect to the sub-problem it solves. This is because they are based on the MCTS algorithm, which offers asymptotic optimality if the rollout process is able to sample the entire action space [235]. Our rollout heuristics in (4.4) and (4.5) ensure that all possible rollout sequences have a non-zero probability of being selected. The proposed rollout heuristics are likely to take actions that are not yet observed, taken by other team members, and are near the vessel. Consequently, we achieve faster convergence in the two sub-problems.

Since the two steps are independently optimised, the overall optimality of Problem 1 is not assured. We argue that the overall solution is practically feasible, even close to a true optimal joint solution. First, we observe that there only exists one unique float trajectory that passes through a given POI in the oceanic flow field as illustrated in Figure 4.2. Intuitively, no float trajectories cross each other. This is because ocean currents are *incompressible* in that the sum of all incoming flows to a point equals the sum of all outgoing flows [236–238]. The observation implies that a truly optimal solution is likely to include those float trajectories passing a large number of POIs. Note that the first step in our framework is to find a set of float trajectories that maximises the number of POIs.

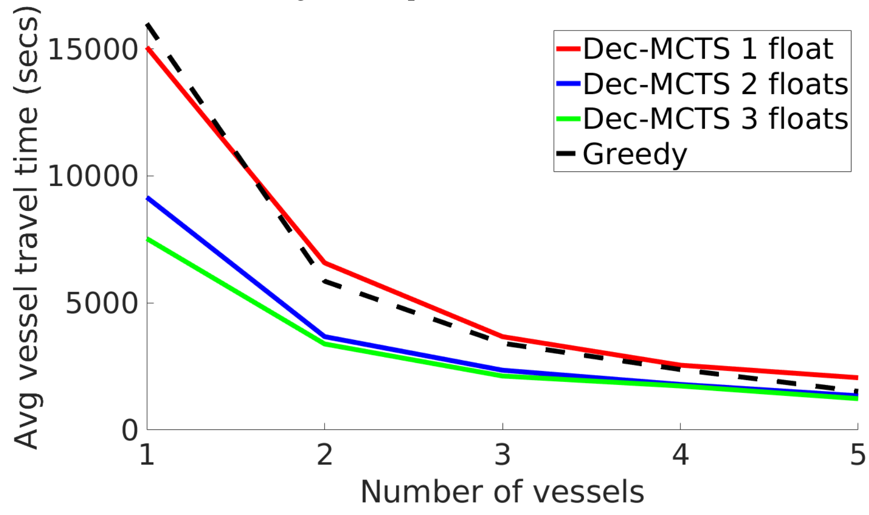
One immediate improvement to the framework is to expand the search space by relaxing a number of constraints we imposed in the first step. For example, varying the lengths of the float paths and planning over other sets of pick-up drop-off actions besides the minimal length POI set.

4.4 Simulation

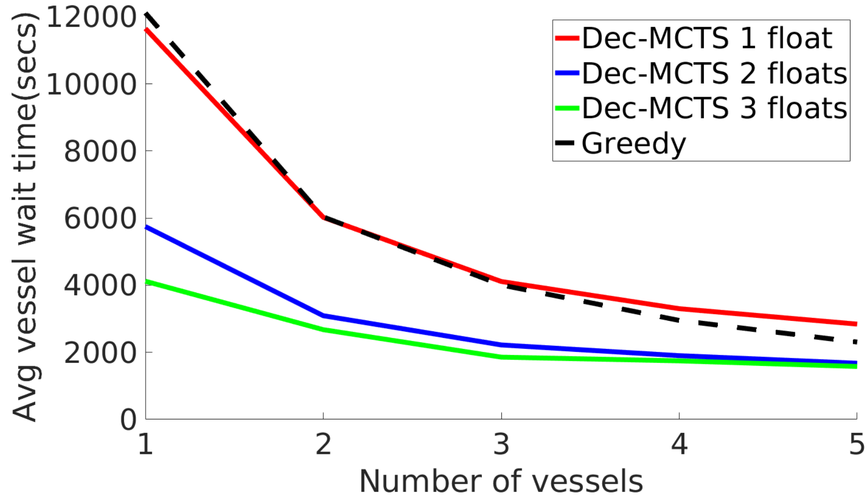
In this section, we present simulation results that validate our proposed framework. We compare its performance against a *sequential greedy scheduler* [102], commonly employed in multi-agent planning applications. We demonstrate our method's ability to produce higher-quality results. Additionally, we briefly discuss the computational requirements of our approach.



(a) Averaged makespan in simulated scenarios



(b) Averaged travel time per vessel



(c) Averaged total amount of time the surface vessels spend waiting for floats

Figure 4.4: Comparing the performance of the proposed framework against sequential greedy method (in dashed black line) for makespan, average work time and waiting time for an increasing number of surface vessels. Three float assignments are shown: one float (in red), two floats (in blue) and three floats (in green) for each surface vessel.

Parameter	Value
Number of ASVs	1-5
Number of floats per ASV	1-3
Maximum flow field velocity	$1m/s$
Maximum surface vessel velocity	$5m/s$
Number of POIs	20
Computation time	60 seconds
Workspace size	2000x2000 meters
Makespan tuning parameter	0.000001
Number of sampled pick up and drop offs	10000
Float trajectory length	5 minutes

Table 4.1: Simulation parameters in Sec. 4.4

4.4.1 Simulation setup

Simulated experiments were performed to empirically show the suitability of our method to solve the MVMF problem. These experiments were performed on simulated flow fields with the parameters shown in Table 4.1. These parameters were selected to be similar to the conditions we expected to experience during field trials at Jervis Bay.

In the simulations, the maximum flow field speed is $1ms^{-1}$, the speed of the surface vessel is $5ms^{-1}$, the number of POIs is 20 and the size of the workspace is $2km \times 2km$. We then empirically compare the performances over a set of different float assignments for an increasing number of surface vessels. The simulations were executed on a desktop computer with Intel i7 2.60GHZ CPU and 16GB RAM. Each simulated experiment was performed 50 times, and the averaged results are shown, with a computation time limit of 60 seconds for each trial. This time limit was selected to allow the system to be feasible for use in the field in a near real-time manner of a short planning window followed by execution of the planned mission.

4.4.2 Simulation results

The compared sequential greedy algorithm works as follows. It sequentially allocates drop-off and pick-up action pairs to the vessel that most improves the joint utility. The sequential greedy algorithm can very quickly allocate the actions and produce valid joint schedules. However, the

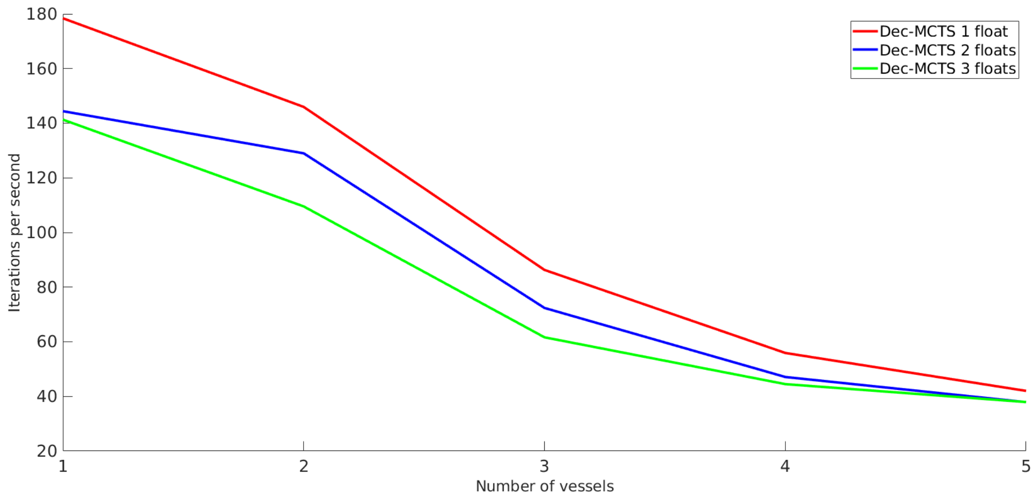


Figure 4.5: Average number of iterations per second for the DEC-MCTS planner within the MVMF framework. The average is taken over 10 runs, and the specific number of iterations is highly dependent on the specific problem and implementation; it should only be used as an indicator of relative computational complexity. These results indicate that incorporating additional floats has less impact than increasing the number of surface vessels. However, DEC-MCTS is asynchronous, so a better parallel implementation should allow better scaling with the number of vessels.

algorithm is not able to utilise multiple floats and can only produce potentially sub-optimal schedules with no further improvement. This is partly because no two floats can be deployed at the same time. Intuitively, the dropped float must be picked up before deploying another.

In Figure.4.3, a visual comparison between the schedule for 2 vessels generated from the surface vessel scheduler (Figure 4.3a) and the sequential greedy (Figure 4.3b) is shown. Both the sequential greedy and our scheduler distribute the float actions between the vessels evenly. However, as the sequential greedy algorithm cannot utilise multiple floats, its best solution is for each vessel to drop-off and pick-up a single float sequentially. I.e., $\boldsymbol{\phi}_i = D_{i,1}P_{i,1}D_{i,1}P_{i,1}\ \forall i \in \{1, 2\}$. In contrast, our scheduler can utilise multiple floats and can consider deploying floats in *parallel* when valid. The benefit of such deployment strategy is shown in Figure 4.3a with the vessel schedules, $\boldsymbol{\phi}_1 = D_{1,1}D_{1,2}P_{1,2}P_{1,1}$ (blue) and $\boldsymbol{\phi}_2 = D_{2,1}D_{2,1}P_{2,1}P_{2,2}$ (green), where the vessels are shown to collectively drop-off and pick-up floats that leads to less wait time and travel time. The parallel deployment strategy is possible because such actions can be considered when building the search trees.

Figure 4.4 shows that our method outperforms the sequential greedy approach in the following metrics: makespan (i.e., the time difference between the team starting and finishing the schedule), amount of vessel travelling time (i.e., the average time vessels spend travelling to drop-off floats), and the overall wait time (i.e., time vessels spend waiting for floats to arrive for pick-up). In Figure 4.4a, we show that the greedy approach works as good as the case with one-surface vessel and one-float combination. However, as the number of floats increases, our method quickly outperforms the greedy in that the overall mission time is shorter.

For higher numbers of floats, our method produces schedules where the surface vessels travel less to complete the mission, as shown in Figure 4.4b. Surface vessels can complete the same amount of work with shorter travel distance because our method utilises the extra floats by dropping them off at nearby drop-off locations. Furthermore, the proposed framework finds a high-quality schedule that reduces the travel distance and wait time, as also illustrated in Figure 4.4c.

4.4.3 Computational evaluation and implications

In this section, we briefly discuss the time it takes to compute solutions for the baseline and our method which can impact the applicability to real-time fieldwork applications with respect to the time to compute solutions.

Finding a good solution for the first problem of sampling the flow field and selecting candidate float trajectories can be solved quickly given a reasonable selection of parameters. The computation length depends on the number of sampled pick-up and drop-off locations, the length of the float trajectories, and the number of iterations used in the trajectory selection MCTS. The trajectory length and number of samples were found empirically. For operational limitations, we determined a satisfactory level by observing a slowdown in the convergence speed of the trajectory selection MCTS within a few seconds. We assumed that additional computation would yield limited additional optimality. For problems of the scale in this thesis, we observed that the sampling and trajectory selection could be completed satisfactorily within a few seconds of computation.

The sequential greedy baseline finds a solution in a fairly consistent amount of time, and it takes under 1 second per surface vessel to compute an allocation for the problems we considered. However, this comes with the aforementioned limitation of not utilising additional floats and having an unbounded suboptimality.

Recall that MCTS and Dec-MCTS are anytime algorithms capable of finding a quick solution and improving it interactively. Therefore, analysing the number of iterations per second is more relevant. Figure 4.5 shows the average number of iterations per second for the DEC-MCTS planner within the MVMF framework. The averages shown here are taken over 10 runs, and the specific number of iterations highly depends on the problem and implementation. However, these numbers are useful for indicating relative computational complexity.

These results show that increasing the number of surface vessels impacts computational complexity more than incorporating additional floats. Although DEC-MCTS is asynchronous and parallelisable, there is still an overhead of communication between vessels and the additional complexity of tracking more actions from other agents. Even with this overhead, the complexity increases at an linear rate rather than exponentially with the growth of the size of the *joint space* of the robot team. In contrast, adding more floats has a limited impact because deploying them in parallel is restrictive (depending on the length of allowed surface loitering time and vessel speed), so the depth of a rollout is minimally impacted.

While determining the precise number of iterations for a 'good' solution is difficult, we empirically observed that 1000-2000 iterations were sufficient to find reasonable solutions for the problems addressed in this thesis. We bounded the computation time to 60 seconds to reflect the desire to use our method in a near realtime manner and balance finding sufficiently good solutions.

Overall, these simulation results validate our claims in Section 4.3.4 that the proposed framework is computationally efficient and it finds a solution over large state space given practical computation time. With this property, we argue that the framework can be used in a *plan-as-you-go* manner, where a new plan is generated when the flow field and the set of POIs change over time. Furthermore, the results demonstrate that our method can reduce the costs of operating a team of vessels and floats due to the reduction in makespan (that allows more missions to be deployed in a given time period), reduction in travel time that means less operating cost of vessels due to wear and tear and, the vessels waiting less.

4.5 Online multi-vessel multi-float flow field problem field trial

In this section, we showcase the practical application of our method by trialling it in the field on real-world hardware and in an oceanic environment. We begin by explaing the field trial setup and the additions required to deploy our method in the field. Next, we discuss the data-gathering process. Finally, we examine the results and discuss new insights discovered.

4.5.1 Field trial setup

In this work, we defined the MVMF Problem 4.3 and proposed a hierarchical solution. The largest and untested assumption in our work is assuming a fully known time-invariant flow field. We assumed that this is a reasonable assumption because ocean currents vary slowly and are driven by prevailing meteorological conditions such as tides and wind. Typically, these processes occur on the timescale of hours, while missions can be designed to last minutes to a few tens of minutes. In other words, it is approximately static. Another practical limitation of our method is the requirement of a dense flow field model for planning the operations due to the float dynamics and our desired application. To plan trajectories for the floats that are capable of observing the bathymetric features represented as POIs, and due to the complex bathymetry of reefs and shorelines, we require spatially dense (vectors at least every meter) over a small-scale (hundreds of meters) model of the flow field. Typical flow field models produced by oceanographic sources are spatially sparse (vectors every hundreds of meters) due to their large scale (hundreds of kilometres). These flow fields are not suitable for our application because the under-actuation of the floats necessitates an accurate knowledge of the flow field for predicting the movement of the system. The unavailability of such a model means that we need to produce a local estimate of the flow field rather than rely on existing sources of information in order to apply the planner in practice.

To gain a deeper understanding of the limitations of the fully-known time-invariant flow field assumption, we introduce a process for estimating the flow field locally, which is then tested with real-world experiments at Jervis Bay. To produce the local dense flow field, we use the Gaussian process (GP)-based expectation-maximisation (EM) method developed by [238] because their approach allows estimation of a dense oceanic flow field from sparse point measurements. The point measurements we use for estimation are the deployment and retrieval locations of the floats or subsampled GPS readings from drifters. The GP method works by exploiting the incompressibility of ocean currents to deal with the underdetermined nature of the problem. The problem is underdetermined because infinitely many candidate flow fields could have generated the same measurements. By exploiting incompressibility, the search space can be reduced so that a feasible solution can be computed. There are two open questions that make the applicability of this method to our problem uncertain. First, the method was tested on sparse GPS point measurements of gliders instead of more dense GPS measurements of drifters. The sparse GP-based method might be not applicable for this much data. Second, the GP method was shown to work for gliders in deep ocean environments. Our field trial takes place in regions with much shallower water (around 10-25 meters depth), near shorelines, and near reefs. It is not yet known if the method will perform as well in this new environment.

4.5.2 Sea trial operations

In this section, we discuss the technical approach that was taken during sea trials to assess the applicability of the flow field GP estimator to real-world oceanic environments similar to our expected operating areas.

To gather this data the drifters or floats need to be spread over the workspace. The size of the workspace and thus needed triangle is limited by both the area that needs to be surveyed by the imaging floats and the communication range of the drifters. The communication range is important because the drifters communicate their locations blindly without caching. If the vessel is outside the communication range, we partially lose the trajectory information. Once the drifters are deployed, they are left to float along with the current for a fixed period of time, then retrieved by the boat. A timely retrieval is important so as not to lose the hardware in the ocean. The selection of the drift length is important because it needs to be long enough to gather sufficient data while not being so long that the data gathered is no longer relevant due to the changing flow field. For our tests, it was found emperically that around 10-15 minutes of drifting combined with the time taken to deploy and retrieve was sufficient. For estimating a flow field, we observed that shorter drifts produced flow fields that were too noisy and longer ones captured changing fields, so their time series had to be cropped; otherwise, their estimations were not useful.

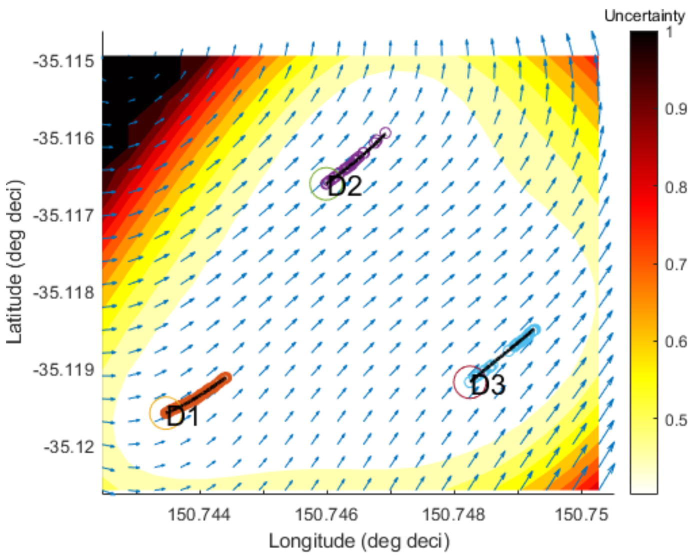
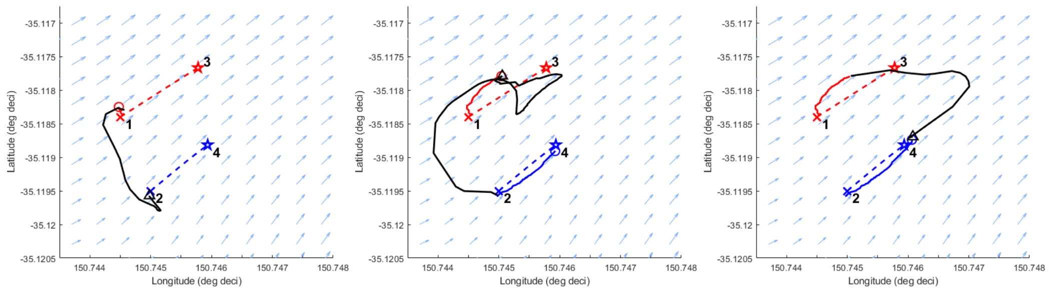


Figure 4.6: Estimated surface ocean current data for a 390m x 390m oceanic environment. Current estimated from three drifter trajectories deployed for 600 seconds in a triangle formation. The colourmap represents the trace covariance. The resulting drift data is estimated using GP-based estimator presented in [238].

The GP-based flow-field estimator then uses the gathered trajectories from the drifters to build an estimate of what flow field best fits the observed trajectories. This estimate and fit were used to determine what length of drifting was sufficiently long for further trials. The GP method depends on a set of hyperparameters, which can require retuning. The tuning process during sea trials uses a grid search over a range of possible sets of parameters and chooses the best set based on the error between the projected trajectory from the estimated field and the observed field. This range of hyperparameters was selected based on a much larger grid search of parameters over historical data, which then informed the smaller grid search over the tuning parameters to evaluate during flow field estimation in the field. Similar to the drift trajectory length selection process, we used the ability of the estimated flow field to explain the trajectory observations to measure the quality of the flow field estimate. We assume the model is accurate if an estimated flow field explains the drift well.

4.5.3 Sea trial results

Twenty sea trials were conducted over five days at two sites (Murray Beach and Nelson Beach) inside Jervis Bay, Australia. We present the most representative and interesting data gathered from these trials.



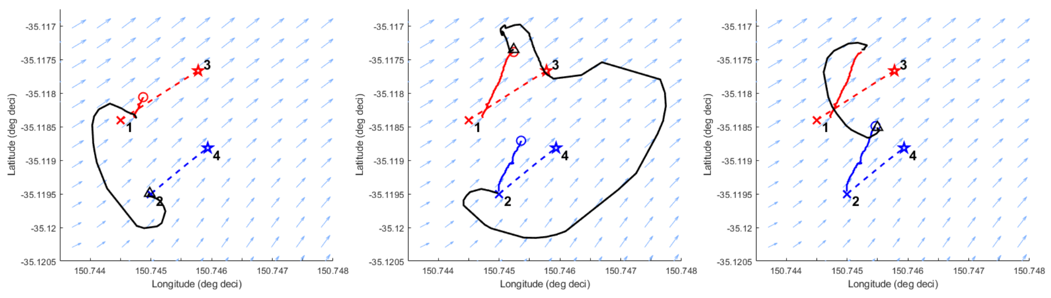
(a) Attempt 1, time: 145 sec (b) Attempt 1, time: 660 sec (c) Attempt 1, time: 790 sec

Figure 4.7: 1 vessel 2 drifter MVMF schedule, planned over the estimated flow field (blue quiver) shown in Figure 4.6, executed one hour after the estimation. Each subfigure shows a snapshot of the *Kimbla* (triangle) moving from one scheduled action to another. For clarity, only the *Kimbla* trajectory (black) associated for that snapshot is shown. The sequence of positions the *Kimbla* is to visit are determined a priori and are enumerated from 1 to 4; marked with either a drop-off (x) or a pick-up (start) action. The actual drifter trajectory (red and blue solid line) is compared against the expected trajectory (red and blue dashed lined). The drifter locations at each snapshot are marked with a circle. The *Kimbla* cut through the red drifter path at both (a) and (b) before waiting in the middle of the workspace to ensure communication to all deployed drifters. At time 600 sec, the *Kimbla* moves to position 3 to pickup the drifter before detouring to the actual drifter location and then picking up the drifter at position 4. The detour resulted in a tardiness of 60 sec for position 3 and 45 sec for position 4.

The trials were conducted aboard the MV *Kimbla*, an 18-meter long vessel from which we deployed a heterogeneous team of under-actuated robotic *floats* [228–230] as shown in Figure 4.1. The University of Sydney developed these robots. The two types of robots we used are a passive *surface drifter* type float, which has no actuation, and a bottom-following benthic imaging-float which is capable of diving to a controlled depth. For both systems, the motion is mainly driven by the ocean currents, which allows interaction with the environment. Both types of robots are equipped with an IMU, GPS, and communicate their state information over LORA. The GPS position information is then filtered by a Kalman filter to reduce noise, During these trials, one imaging float and three drifters were utilised to gather the current estimation data. A visualisation of the drifter deployment for flow field estimation is shown in Figure 4.6.

4.5.3.1 Trial 1 – drop-pick evaluation

We evaluate the consequence of the quasi-static assumption on the MVMF schedule with a simple drop-off/pick-up mission using the *Kimbla* with two drifters. The *kimbla* operation consists of moving from location to location in the schedule and deploying or retrieving drifters as needed. If

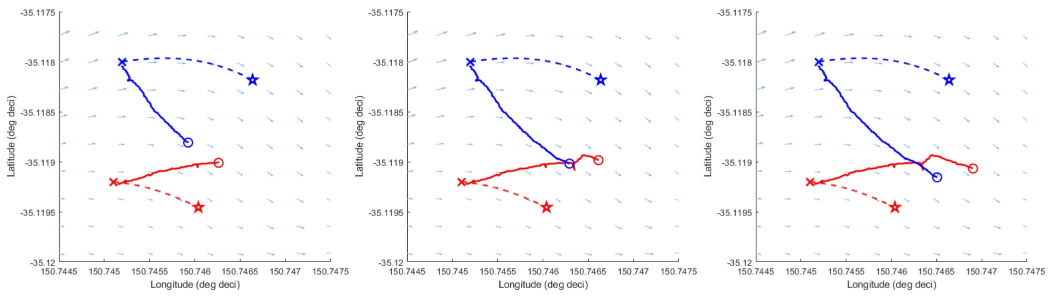


(a) Attempt 2, time: 230 sec (b) Attempt 2, time: 680 sec (c) Attempt 2, time: 855 sec

Figure 4.8: Same mission as Figure 4.7, except the schedule was executed two hours after estimation. To avoid the effect the *Kimbla* has on the drifters, this mission was executed with slower vessel velocity and having the *Kimbla* travel around the drop-off and pick-up locations. Both drifters however travelled more northerly compared to the expected drifter trajectory, which implies the ocean current have changed. Again, the *Kimbla* visited position 3 at time 600 sec before detouring to the actual drifter location then picking up the drifter at position 4. The detour resulted in a tardiness of 80 sec for position 3 and 25 sec for position 4.

a drifter is not at the retrieval location when expected, the new retrieval location is set to the current GPS position. The boat then moves to retrieve the drifter at its new location, and the mission continues as expected. We execute the same MVMF schedule twice: one hour after estimation and two hours after estimation. The evaluation compares the expected drifter trajectory from the schedule with the observed drifter trajectory from both schedule attempts. If the quasi-static assumption holds, then the expected and observed trajectories should be similar. For the evaluation, we arbitrarily choose two drop-off locations in the workspace and define the corresponding pick-up locations and expected drifter trajectory by forward integrating across the estimated flow field for 600 seconds. We also define the action sequence a priori as parallel deployment as shown in Figure 4.7.

The snapshot of executing the first schedule attempt is shown in Figure 4.7. The *Kimbla* deploys the drifter at point 1 and 2, before waiting at the middle of the work environment to maintain communication between the *Kimbla* and the drifters. After 600 seconds, the *Kimbla* attempts to execute the scheduled pick-up actions while compensating for any discrepancy between the expected and observed result. The blue drifter appears to closely follow the expected trajectory. However, the red drifter trajectory is translated and slower than the expected result. We suspect that the *Kimbla* may have created wakes near the red drifter that stalled and displaced it. We see that the *Kimbla* cuts across red drifter's expected trajectory in both Figure 4.7a and Figure 4.7b. The delay caused by the wake forces the *Kimbla* to deviate from its schedule to retrieve the drifter,



(a) Mission1, time: 1347 sec (b) Mission1, time: 1795 sec (c) Mission1, time: 2084 sec

Figure 4.9: Snapshot of two drifter trajectory (red and blue) crossing. Expected trajectory and estimated flow field shown for comparison.

which by then the blue drifter passes over the expected pickup location before it is retrieved. This result brings up an interesting challenge when planning for MVMF schedule:

Remark 1 (Disturbance due to surface vessel)**.** *Recklessly operating the surface vessel near the float robot or its expected path can disrupt the intended float trajectory, which can negatively impact the overall quality of the MVMF schedule.*

This remark applies for both drifter and shallow diving float implementation. Though the disturbance on the drifters due to the *Kimbla* is more obvious for our small-scale trial, it nonetheless should be avoided for general cases as the ocean current is a chaotic system. In particular, care needs to be taken when deploying drifters for flow field estimation as they deploy around the same time scale as our trial missions.

Snapshot of the second schedule attempt is shown in Figure 4.8. After learning from our previous attempt, we were cautious to operate the *Kimbla* without creating wake near drifters or its expected path by travelling around the drop and pick location. While there was no evidence of the *Kimbla* affecting the drifter trajectory, we observed they both drifted more northerly than the expected trajectory. This implies that the ocean current has changed between the first and second attempt. However, it remained reasonably static between the estimation step and the first attempt as described in:

Remark 2 (Valid time window for ocean estimation)**.** *The quasi-static flow field assumption holds true for real-world implementation for about an hour.*

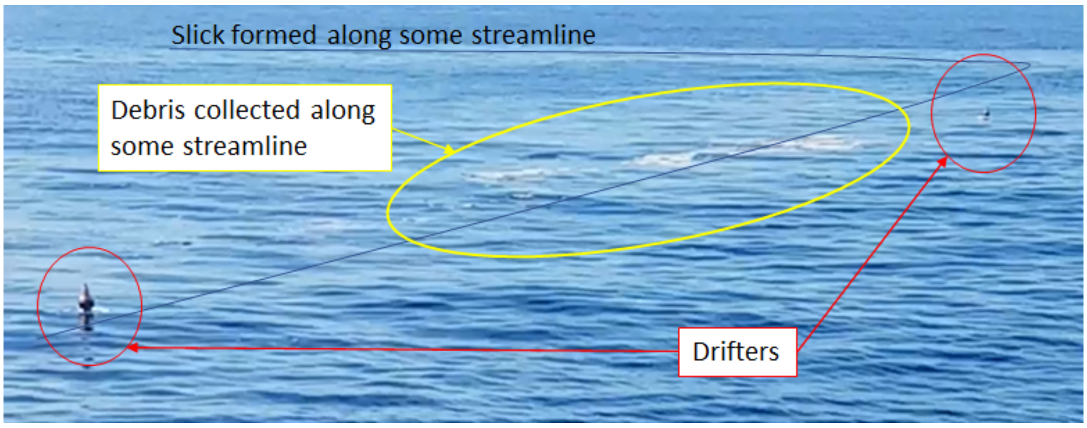

Slick formed along some streamline
Debris collected along some streamline
Drifters

Figure 4.10: Drifters on the ocean surface in an environment consistent with Langmuir Circulations

4.5.3.2 Trial 2 – Langmuir circulation

During our trial, we also observed a special flow field structure that contradicts the planar incompressibility assumption made for estimating the flow field. Two deployed drifters that were expected to move parallel to each other instead moved towards each other until, eventually, their trajectories crossed. The snapshots of this case are shown in Figure 4.9.

As both drifters were deployed relatively close together, mere minutes apart, the trajectories crossing implies that the oceanic streamlines cross, which is impossible in a planar incompressible flow field. One hypothesis is that the drifters were sucked into a Langmuir circulation cell. That is, there is a pair of corkscrew vortex rotating about the direction of the flow that draws nearby water into a stream and pushes it down underwater. This oceanographic process occurs in the presence of wind shear and when wind and waves move in a similar direction. We were also able to observe this current phenomenon. Figure 4.9 shows sea debris, and slick were collected along a streamline, and the floats were observed to be riding these streams. The existence of such a flow structure invalidates the assumption we use for our estimator: that the flow field is planar and incompressible.

Remark 3 (Violation to assumption on incompressibility)**.** *There exist oceanographic processes that may violate the incompressibility assumption for explicit 2D flow field cases near the ocean surface.*

4.5.4 Potential approaches

During our field trial, we have identified various challenges that can be addressed to improve the overall performance of executing a MVMF schedule in a difficult real-world environment. In this section, we reflect on these challenges and discuss potential approaches to address these challenges.

Disturbance due to surface vessel

From Remark 1 and from the second schedule attempt shown in Figure 4.7, we need to operate the surface vessels with caution when near a drifter or shallow-depth float robot. There are several operation-based approaches to mitigate the effect of surface vessels on floats, such as delaying departure to the next position or reducing surface vessel speed when nearby floats. One possible algorithmic solution is to plan the surface vessel trajectory such that, topologically, it does not intersect the drifter trajectories nor come near the floats during deployment. Such an approach allows the surface vessel to operate at max capacity while guaranteeing it will not influence the float trajectory.

Valid time window for ocean estimation

As stated in Remark 2, Based on observed data the quasi-static flow field assumption is reasonable for approximately an hour. However, such a window of operation limits the scope and size of the mission due to the slow speed of the floats. In addition, the ocean flow field appears to make sudden changes. That is, we can not use the drift information from the first schedule attempt to predict the flow field for the second schedule attempt. This challenge motivates a time-varying flow field estimator and planning a MVMF schedule over it. However, the time-varying problem is intrinsically difficult. Another approach could be to integrate the estimation step into the MVMF schedule planner and have the system be closed-looped. This approach has the added bonus of being able to autonomously recall floats when the environment becomes too unstable mid-mission.

Violation of incompressibility assumption

From Remark 3, it is clear that we do not yet fully understand the ocean dynamics enough to make robust flow field estimations. For instance, the GP-based estimator presented in [238] does not account for wind, tide, or bathymetry, any of which impacts both ocean surface and depth flow field. This challenge motivates a more comprehensive flow field estimator that accounts for other oceanic factors. The flow field estimator assumes a valid 2D incompressible flow field. External disturbances from wakes or 3D oceanographic phenomena invalidate its assumptions and will lead to incorrect estimations. So, care needs to be taken to ensure that the system operates under conditions in which these assumptions hold.

4.6 Replanning field trial

During the trials, we experienced situations where the floats took a different trajectory than the one estimated by the calibration measurements. The cause is ocean currents changing in the time between calibrating for the flow field estimation and the observation deployments. To validate this is the cause, we deployed the drifters and then shortly after redeployed them in the same area and compared the expected and observed trajectories. This process can be seen in Figure 4.11. From these tests and the previous field trials it can be seen that a system that is able to adapt to the changing ocean currents is desirable. One approach to online planning is to continuously update the flow field estimation using new drift information obtained throughout the MVMF missions. Each time a float resurfaces or when a drifter is ready to be retrieved, the observed drift information would be included in the flow field estimator and replanned with the new estimated flow field.

Using this replanning strategy, we performed another set of field trials. Figure 4.12 shows the field trial to evaluate the planner with replanning. The schedule for the floats is initially planned over an initial flow field using the estimation from the calibration dive. New float drift information from the deployment is used to re-estimate the flow field, and the schedule is replanned over it. The interval between each float dive was 30 minutes. The figure shows the results of sequential replanning, changing the plan in response to the change in the flow field between each deployment.

These results demonstrate that the MVMF method we developed works if the quasi-static flow field assumption holds. However, if the assumption is invalidated due to the operational environment then with a simple modification online replanning can be used. It is able to be used in an online replanning manner due to its design making it quick to compute and scalable.

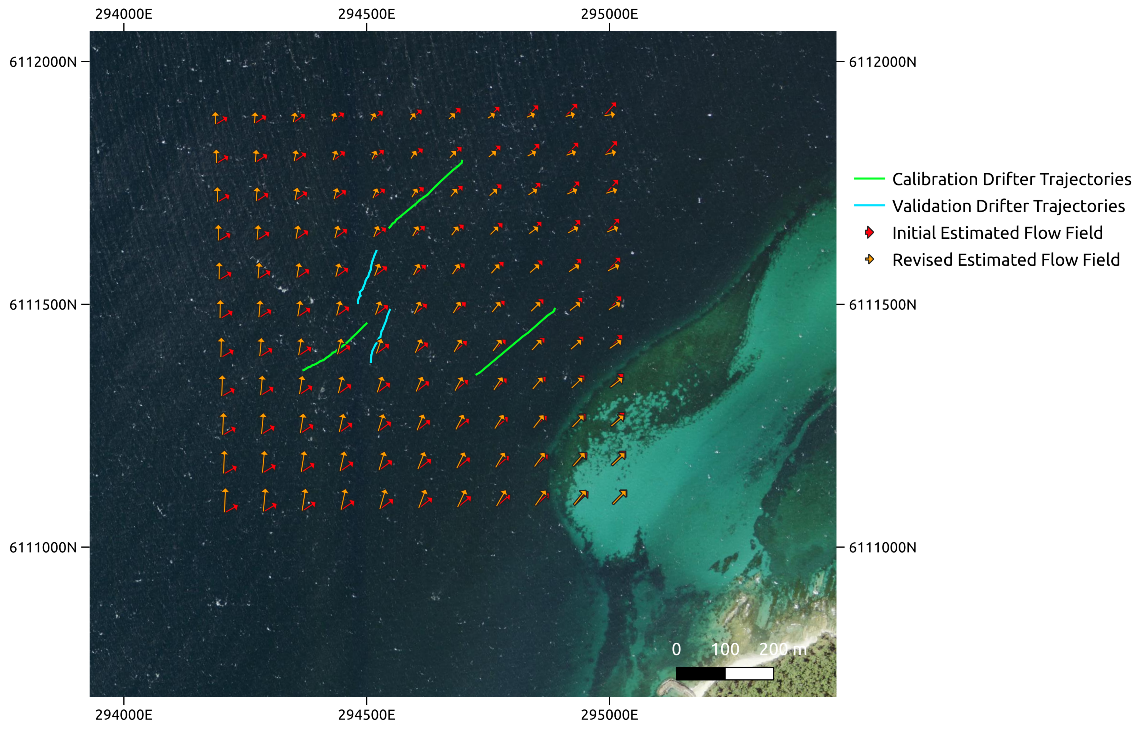
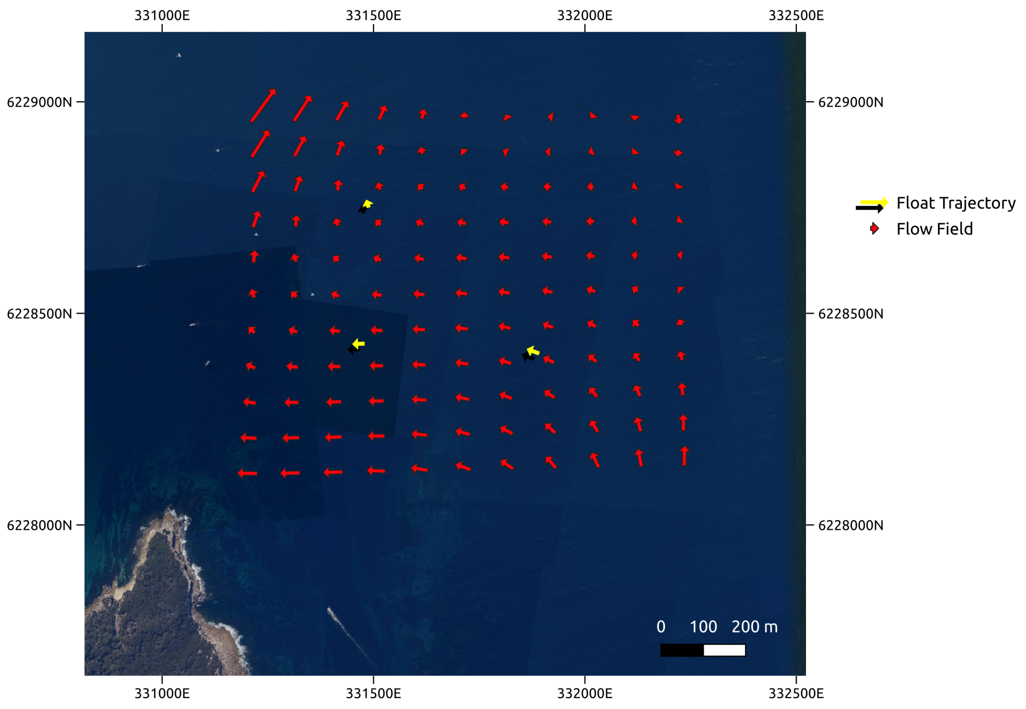
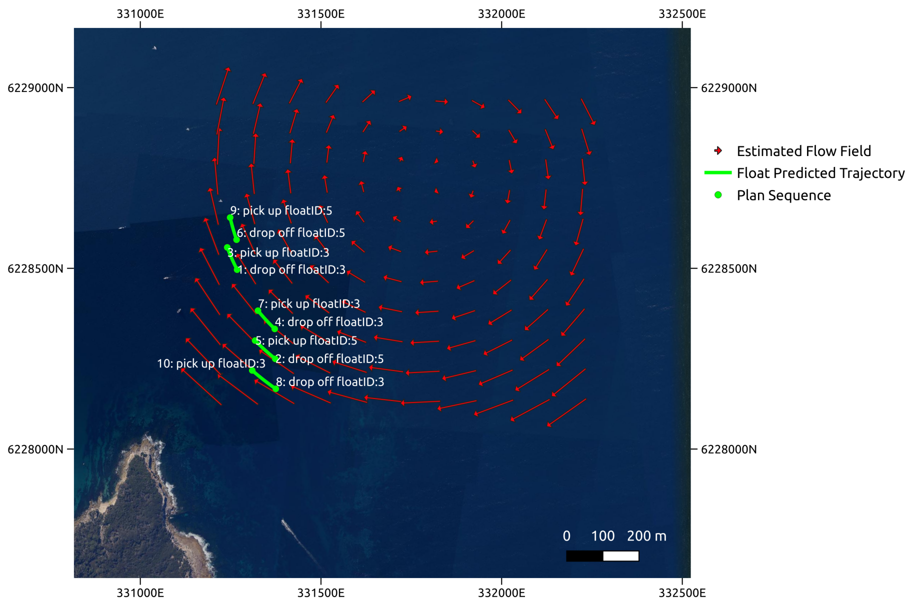

294000E
294500E
295000E
6112000N
6111500N
6111000N
Calibration Drifter Trajectories
Validation Drifter Trajectories
Initial Estimated Flow Field
Revised Estimated Flow Field
0
100
200 m

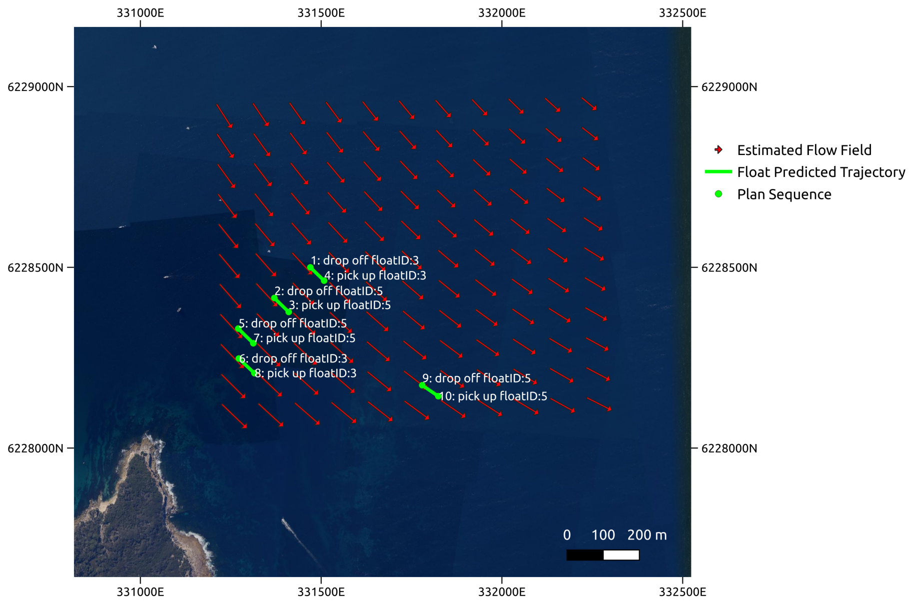
Figure 4.11: Example of ocean currents changing with time. The estimated ocean current using the calibration trajectories (red arrows) is different to the current estimated using the validation trajectories (yellow arrows). The two trajectories were taken hours apart.



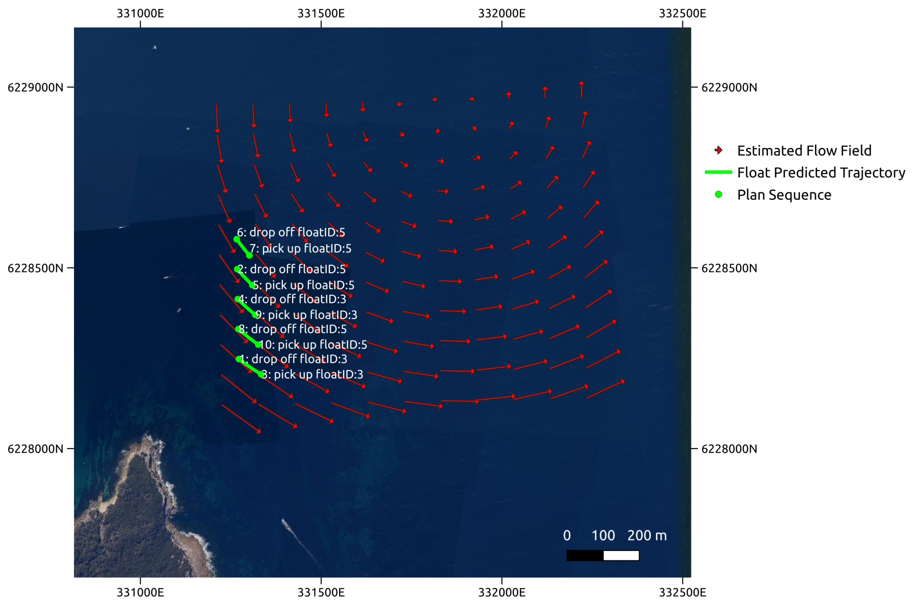
(a) Initial deployment and plan



(b) Replan 1



(c) Replan 2



(d) Replan 3

Figure 4.12: Field trial to evaluate the planner under real-world mission conditions for a 1-float 1-vessel case. planner finds the initial schedule (yellow) over the perceived flow field (red). After a float resurfaces, its drift information is considered to re-evaluate the flow field and the appropriate new schedule (green).

4.7 Summary

This chapter presented a solution for scheduling a heterogeneous team of information-gathering vessels operating in the ocean. We demonstrated how a generic vehicle routing formulation would not be feasible, and then we formulated the more specific multi-vessel multi-float problem. In this problem, the goal is to observe points of interest by deploying bottom following floats controlled by the ocean current and picking them up when they surface. The floats cannot be left unattended on the surface for longer than a prescribed period, and the ocean current dynamics are chaotic, making finding feasible schedules difficult. We address this difficulty by assuming the ocean is quasi-static and planning short operations. To plan the operations, we use a hierarchical planner that solves smaller subproblems in a decentralised manner. We empirically demonstrate our method's scalability and effectiveness through simulation. Next, we tested our planner on real hardware in field trials, and through these field trials, we observed potential future research directions. Finally, we tested a replanning extension to our algorithm in the field to verify if the idea improved the approach's applicability to real-world operations.

Through this chapter, we demonstrated a solution to scheduling operations for problems with stochastic constraints. In the MFMV problem, the stochasticity of the constraints is created by the ocean currents. This problem was addressed through understanding the physical process that causes the constraints to change and, thus, plans to become invalid. We were able to make plans feasible by restricting operations to a short time horizon that is smaller than the larger timescale of ocean current changes. This quasi-static assumption is beneficial because it makes the operations feasible and avoids the need to predict the future of chaotic ocean currents.

In the next chapter, we examine scheduling problems where the tasks that need to be scheduled are uncertain and are driven by a stochastic process. In stochastic constraint problems, the tasks are fully known, and the plan to service the tasks can become invalid. Whereas, in stochastic task problems, the tasks that need to be completed are uncertain, but a plan does not become infeasible during execution. In these problems, the focus is less on creating methods that can make plans feasible but instead on finding techniques to deal with completing potential future tasks.

Chapter 5

Stochastic Tasks

In this chapter, we focus on multi-agent scheduling problems with stochastic tasks. Tasks are the work the agents complete by performing actions that are subsequently rewarded by the objective function. The uncertainty in the formulation of the task defines stochastic tasks. This task uncertainty can take multiple forms, such as not knowing 'which' tasks need to be done, 'when' the task will be completed, or 'how' to complete the task due to it having stochastic costs.

This is the third component of our probabilistic scheduling framework. Unlike previous problem classes that try to optimise with uncertain rewards for actions or try to find strategies to make a specific plan feasible, it is unknown what work will be needed. Any feasible plan, once created, will remain feasible but may be of low quality when considered in hindsight with the full sequence of tasks defined. Stochastic task methods aim to avoid this regret by making decisions that consider what could be required in the future.

We examine the class of scheduling with stochastic tasks by solving dynamic warehouse order picking. In this problem, teams of logistics workers (agents) in an order-picking warehouse servicing dynamic, uncertain real-time orders throughout the day. Similar to previous work, we develop a solution to the dynamic vehicle routing problem by partitioning the workspace into smaller regions. The main novelty of our approach is modifying the cost metric used to form the partitions to be more applicable for warehouse order picking. Analytically, we show that to maximise throughput, the work in each region needs to be balanced. We then present a method to compute this measure of work in each region and how to balance it between partitions iteratively. Finally, we evaluate our results in simulations of a typical warehouse picking environment and compare them against a less specific partitioning algorithm and the naive method performed in practice. Our results outperform the tested algorithms and demonstrate the effectiveness of our method.

5.1 Overview

Stochastic task scheduling problems are a class of problems where the tasks that need to be scheduled are not fully known a priori. Three main components of a task that need to be known to fully schedule them. The start time, the length or end time, and the resources it costs. A stochastic task is one where the arrival time of the task is not known exactly, and they are driven by some stochastic process (either observable or not). The processing time or the length of time it takes to complete a task can be similarly probabilistic. For the stochastic resources, the person or machine that is capable of performing the task could be unavailable, and the planner needs to account for this uncertainty. Real-world examples where this class of problem occurs are hospital emergency departments [239], manufacturing [240], delivery of goods and servicing e-commerce orders in a warehouse [241].

For this chapter, we focus on the warehouse order-picking problem. A central operation in warehouse logistics is *order picking*, where items are collected from their warehouse storage locations and then packaged to fulfil customer orders. Order picking is an on-demand process that can benefit greatly from optimisation. Since it is not possible to know what items customers will order, the problem is inherently a stochastic task scheduling problem.

Improving order-picking efficiency is also important in terms of cost savings. Because manual order picking accounts for 55% to 70% of warehouse operation costs [242], a modest improvement in efficiency can yield considerable economic benefits. Manual *picker-to-part* order picking systems are the most common in practice (up to 80% adoption) [243], but automation technologies, including multi-robot systems, are also available and used.

In this chapter, we consider the problem of designing an efficient planning algorithm to optimise order picking in warehouses. Our intention is to design algorithms that are applicable to both robotic and manual operations. The rate at which orders arrive is not necessarily regular, so planning algorithms must account for this *dynamic* aspect. Warehouse operations often require a *team* of people or robots (agents), so planning algorithms must also be capable of utilising the cooperation of multiple robots to achieve high throughput. Finding optimal algorithmic solutions, unfortunately, is infeasible for all but the smallest of scenarios because component subproblems such as routing, batching, and allocation are NP-hard [81, 244].

To account for the scalability problem, we propose creating a method that solves the dynamic warehouse picking by partitioning the environment into areas of *equal work* and then using a service policy. According to an estimation of expected work, these partitions are equitable, accounting for an irregular order arrival rate. These partitions facilitate the allocation of work to each robot/agent: When an order of several items arrives, items are considered individually, and work is allocated to the partition that contains the corresponding item. These item demands are then serviced by robots/agents who follow a given policy. This approach of picking orders at the level of constituent items requires a collating process at the depot where items are dropped off, but this is a worthwhile tradeoff since picking is the dominant process in terms of time, complexity, and cost. Our formulation scales well with the number of robots/agents because it avoids the joint optimisation problems involved in batching and allocation and due to efficiencies gained through the distributed design of the partitioning algorithm.

More formally, we create an initial set of Voronoi partitions [245] using *kmeans++* [246] seeding and then use *Lloyds algorithm* [247] to compute an initial spatially balanced/equally weighted power diagram [248]. Subsequently, we iteratively update the weights in the power diagram using a local optimisation algorithm that balances regions according to a heuristic measure of *work* required to service a partition. The measure of work is the expected time it will take to visit a set of items to pick up and then return to the depot to drop off. This metric is computed by sampling orders from a spatio-temporal model of order arrival. Once the partitions have been created, a robot/agent is allocated to each and follows a policy to service a queue of demands allocated to its partition. A benefit of the partitioning method is that our algorithm has the *anytime* property; each iteration produces a feasible candidate partitioning. Additionally, its fast computation speed means that it can be used as a design tool to evaluate the performance of various system characterisations. Typically, warehouse operations seek to maximise the utilisation of robots/agents. Therefore, we evaluate our method operating in a near heavy-load regime (i.e., the order arrival rate is approximately equal to the service rate). The policy and design of our algorithm behave accordingly and exhibit the best performance under these conditions.

The empirical effectiveness of our method is evaluated through simulation of dynamic-order picking performed in a realistic warehouse environment. Our method is compared against a previously not warehouse-specific partitioning-based dynamic vehicle routing solution and a typical process called pick lists that are commonly seen in warehouse operations. Overall, we show that our method is scalable and outperforms the comparative methods that do not leverage the prediction of the work required to service future tasks.

5.1.1 Chapter outline

The remainder is this chapter is organised as follows. Section 5.2 defines the dynamic warehouse ordering picking problem. Section 5.3 provides some analysis showing the throughput is optimised when balanced according to work. Subsequently, we outline the algorithm to create equitable partitions and the service policy. Section 5.4 gives some discussion about the properties of our method, such as its scalability and approximation quality. Section 5.5 presents the results of simulated experiments for empirical analysis of our algorithms. Finally, Section 5.6 concludes the chapter.

5.2 Problem formulation

In this section, we present the problem formulation for the warehouse ordering picking problem. In this problem, given a team of agents and dynamically arriving orders, the goal is to find a policy that maximises the expected throughput of items over a long horizon. We first begin by showing how solving this class of problem as a vehicle routing problem is difficult. Subsequently, we define a queueing-based problem formulation that is more manageable to solve in the following sections.

5.2.1 Vehicle routing formulation

The order-picking problem has been described as a vehicle routing problem. We define warehouse-order picking as a vehicle routing problem and show that this more generic definition can be improved upon by considering the properties of the problem more carefully.

Given a set of n demands $\mathcal{D} = \{1, \ldots, n\}$ and K agents. Where each agent starts at the depot and then visits a subset of demands and returns. All demands can be visited exactly once. Each agent has a maximum capacity of Q, limiting the number of possible demands that can be serviced before returning to the depot. Representing the environment as a complete graph $G(\mathcal{V}, \mathcal{E})$ of which the set of demands are a subset of the overall possible vertices $\mathcal{D} \subset \mathcal{V}$ for notation's sake, we say the depot is vertex 0 and vertex $n + 1$. This allows the construction of routes that go from 0 to $n+1$. The set $\mathcal{E}$ contains the edges between each pair of vertices with an associated cost c_{ij}. Each demand has an associated service cost of $q_i > 0$, with the cost of the depot vertices set to 0 $q_0 = 0$ and $q_n + 1 = 0$. The objective for a CVRP is to find a set of minimal-cost routes that service all the demands, if possible.

Additionally, the time window component introduces a temporal component to the CVRP/VRP; we define a time window as $[w_i^a, w_i^b]$, which defines the start time at node i as w_i^a and needs to be completed before w_i^b. For the rest of the temporal components, we define a service time s_i for each node and a travel time t_{ij} for each edge in the graph. Let M_{ij} be a sufficiently large value to ensure that jobs are completed on time $M_{ij} - max(w_i^b - w_j^a)$. We define the decision variables of the problem. Let x_{ij} be a binary decision variable representing if an edge is taken in the routes. Let y_i be a continuous decision variable representing the accumulated demand of the route that goes from the depot to vertex i, including vertex i. Subsequently, for the time window form, let w_i be a continuous decision variable representing the time at which service starts at node i. Finally, using the above, we can define the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) problem as follows:

$$\min \quad \sum_{i=0}^{n+1}\sum_{j=0}^{n+1} c_{ij}x_{ij} \tag{5.1a}$$

$$\text{subject to} \quad \sum_{j=1,j\neq 0}^{n+1} x_{ij} \leq 1 \qquad i = 1,\dots,n, \tag{5.1b}$$

$$\sum_{i=0,i\neq h}^{n} x_{ih} - \sum_{j=1,i\neq h}^{n} x_{hj} = 0 \qquad h = 1,\dots,n, \tag{5.1c}$$

$$\sum_{j=1}^{n} x_{0j} \leq K, \tag{5.1d}$$

$$y_j \geq y_i + q_j x_{ij} - Q(1 - x_{ij}), \qquad i,j = 1,\dots,n+1, \tag{5.1e}$$

$$q_i \leq y_i \leq Q, \qquad i = 0,\dots,n+1, \tag{5.1f}$$

$$w_j \geq w_i + (s_i + t_{ij})x_{ij} - M_{ij}(1 - x_{ij}), \quad i = 0,\dots,n, j = 1,\dots,n+1, \tag{5.1g}$$

$$w_i^a \leq w_i \leq w_i^b, \qquad i = 0,\dots,n+1, \tag{5.1h}$$

$$y_i \in \mathcal{R}^+, \tag{5.1i}$$

$$w_i \in \mathcal{R}^+, \tag{5.1j}$$

$$x_{ij} \in \{0,1\}. \tag{5.1k}$$

A brief description of the constraints are: Equation 5.1a is the cost-minimising objective, and since the cost is the distance travelled, it is the same as minimising total time. Constraint 5.1b ensures that each demand is serviced only once. Constraint 5.1c guarantees the flow of agents if an agent arrives at node h, then it must depart from this node; it cannot teleport around. Constraint 5.1d is used to cap the maximum number of routes to K, so at least K agents need to be allocated. Constraint 5.1e Constraint 5.1f are used to eliminate sub-tours and to ensure that the capacity constraints are respected. Constraint 5.1g and Constraint 5.1h are used to ensure that a job cannot start before its start time and that travel time between jobs is respected Constraint 5.1k the integer decision variable saying if the edge from i to j is taken. This formulation can be solved in a number of ways, and there is a large body of literature on solving this class of problems. However, these types of solutions do not scale well. Additionally, this formulation currently does not have a concept of stochastic tasks.

Complexity discussion

To extend the CVPRTW problem definition to work with stochastic tasks, the arrival times of the tasks would need to be known. The set of demands $\mathcal{D}$ is unknown, and the time windows they may arrive at are also unknown. Without knowing these, the best you can do is either predict the future (which is infeasible for this problem) or solve the problem using existing demands and then replan as new demands arrive. The integration of newly arriving orders and replanning has been attempted in the literature but has been shown to perform poorly as the problem scales and the arrival rate of orders increases. This breakdown occurs due to the complexity of solving the initial problem each time and/or the plans not being created while considering the potential future arrival of demands.

This is why replanning ignores the unknown future demands and serves what is known here and now. This approach is sufficient for some instances of the problem, but if the rate of dynamically arriving demands is high, the solution quality suffers greatly due to myopia and can be deadlocked. Predicting the future and preemption can be used to alleviate some of this downside if it is possible to model a distribution of future demands. If we had an oracle that could perfectly tell you the upcoming demands, then the stochastic problem would be identical to the deterministic problem. Unfortunately, this is not feasible in practice. However, if we assume that the historical demand data is predictive of future demands, then we can use them to create a policy that will do well considering the wide variety of possible realised futures.

5.2.2 Dynamic warehouse order picking

Suppose we have a two-dimensional warehouse $\mathcal{W} \in \mathbb{R}^2$ with a known layout of the shelves and obstacles $\mathcal{O} \subset \mathcal{W}$ as depicted in Figure 5.1. The warehouse stores a known set of items $\mathcal{I} = \{I_0, I_1, \ldots\}$ and their pickup locations $\mathcal{X}_I = \{\mathbf{x}_0, \mathbf{x}_1, \cdots\}$ are known in advance. We assume that the stored quantity for each item is practically infinite, and there exists only one location to pick up an item $I \in \mathcal{I}$ (i.e., item I_k can only be picked up at $\mathbf{x}_k$). All items in $\mathcal{I}$ have the same weight and size. Items are then dropped off at *depot* $D \in \mathcal{W} \backslash \mathcal{O}$ to be serviced.

We define an *order* as a set of item-quantity pairs to be serviced. The order received at time t is denoted as $\mathbf{o}_t = \{\{I_k^t \in \mathcal{I}, N_k^t \in \mathbb{Z}^+\}\}_{I_k \in \mathcal{I}} \in \mathbb{O}$, where $\mathbb{O}$ denotes a set of all possible orders (i.e., item-quantity pairs). The timed discrete sequence of orders received is denoted as $\mathbf{O} = \mathbf{o}_{t_0}\mathbf{o}_{t_1} \cdots$.

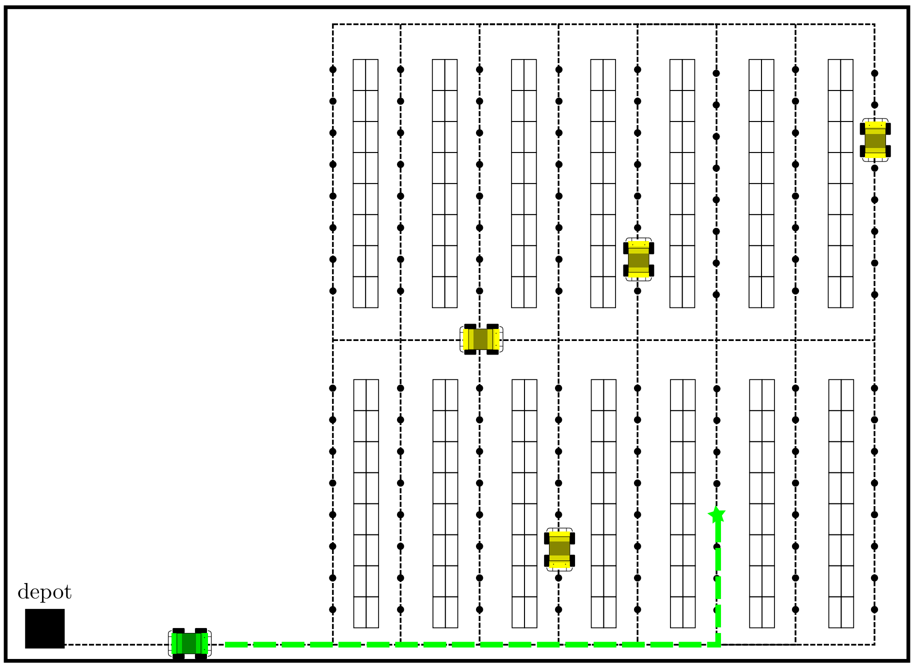


Figure 5.1: Example illustration of a warehouse with robots performing order picking. The black circles are the item locations that robots must visit to pickup an item. The depot shown as a black square is where robots must return to drop off their collected items.

We assume that the interarrival time of orders and the item-quantity pair within orders are both independent and identically distributed. As commonly practiced [249–251],

Order arrivals are modelled as a continuous time Poisson process $\{\mathcal{P}(t)\}_{t\in\mathbb{R}^+}$. The contents of those orders (i.e., item-quantity pairs) are drawn according to item selection distribution ϕ and quantity distribution ψ. Note that the item and quantity distributions can be arbitrary and are often derived based on a historical dataset of orders. An order is considered serviced when all requested items are delivered to the depot D.

Items in orders are added to a first-in-first-out (FIFO) queue $\mathbf{Q}$ defined as a discrete sequence of items, denoted as $\mathbf{Q}(t) = (I^k \in \mathcal{I})_{k\in\mathbb{Z}_{\geq 0}}$, where k indicates the indexes of items (i.e., the very first item added to the queue is denoted as I^0). The queue describes the set of items yet serviced at time t. Therefore, the number of items serviced, denoted as $N(t)$, is $k-1$ where k is the superscript of the next item to be serviced in the queue.

To service the orders received, we deploy m homogeneous robots/agents $\mathbf{R} = \{\mathbf{r}_0, \cdots, \mathbf{r}_m\}$ (henceforth called robots for clarity) where the number does not change. Robot $\mathbf{r}_i \in \mathbf{R}$ is capable of carrying R_c items at a time and moves at a constant speed R_t. The time it takes to pick up and drop off items are R_p and R_d, respectively. A robot $\mathbf{r}_i \in \mathbf{R}$ is assigned a multiset of items from the queue Q called *batch* B_i where the number of less than or equal to R_c. Note that a batch may contain duplicate items. Once robot $\mathbf{r}_i$ delivers all items in the batch to the depot D, it receives a new batch. Given a batch B_i for robot $\mathbf{r}_i \in \mathbf{R}$, a policy $\sigma_i : B_i \to (\mathcal{W} \backslash \mathcal{O})$ is used to complete the batch, where $\sigma \in \Sigma$ dictates a sequence of positions in environment $\mathcal{W}$; the set of policies for all robots $\mathbf{R}$ is denoted as Σ. The sub-route between two adjacent positions is assumed to be the collision-free shortest path, which can be easily found using path planning algorithms like *probabilistic roadmap* (PRM) [211].

We define the warehouse picking problem as how to allocate i.i.d. arriving orders $\mathcal{O}$ and the corresponding items to robots to maximise the number of items serviced. We formally define the warehouse picking problem as follows:

Problem 5.1 (Dynamic Warehouse Order Picking Problem)**.** *Given a set of items $\mathcal{I}$ and their locations $\mathcal{X}$ in a warehouse $\mathcal{W}$ with obstacles and shelves $\mathcal{O}$, and the spatio-temporal process $\mathcal{P}$ that describes the order and the time of arrival, find a set of optimal policies $\Sigma^* = \{\sigma_0^*, \cdots, \sigma_m^*\}$ for service robots* $\mathbf{R}$ *with capacity R_c that maximises the expected throughput of servicing items over a long time horizon, such that*

$$\Sigma^* = \arg\max_{\Sigma} \lim_{t \to \infty} \mathbb{E}\left[\frac{N(t)}{t}\right], \tag{5.2}$$

where robots $\mathbf{R}$ *receives new batches of items from the queue Q.*

5.3 Distributed equitable partitioning

Partitioning of the environment is a well-known approach for solving DVRP problems with infinite capacity [46], and limited capacity [178]; we extend it to work in warehouse order-picking environments. To solve the dynamic warehouse order picking problem, we consider a class of policies that partition the environment $\mathcal{W}$ into a set of m regions $P_1, \ldots, P_m \in \mathcal{M}$, where the pickup locations of items within each partition are assigned to a robot. The key idea is that the maximum expected throughput of servicing items can be achieved by minimising the expected wait time of the orders. This minimisation can be done by servicing batches of orders in each partition at around the amount of time it takes for a new batch to arrive. This arrival rate is dictated by the spatio-temporal process $\mathcal{P}$. The service rate is controlled by the partitions and the policy that robots follow. A robot $\mathbf{r}_i \in \mathbf{R}$ assigned to a partition P that is characterised by a subset of items $\mathcal{I}_i \subset \mathcal{I}$ in the warehouse and the corresponding policy $\sigma_i(B_i)$ that dictates the order of pickups.

The distributed warehouse partitioning problem is decomposed into three sub-problems. We first formally define a service policy and associated cost metric for each robot that evaluates the expected service time for a batch. We then present a sampling-based evaluation of the cost metric given an arbitrary spatio-temporal process. We argue that our framework can work with any type of spatial distribution that describes which items are within an order. Finally, we use the sampling-based cost metric to iteratively find a set of equitable partitions that perform well for the defined service policies for robots.

Finding the expected service time for a batch requires defining the robots' policy to do the servicing. The choice of policy can affect the partitions that need to be created. Our first step is to compute the expected cost of servicing demands using this policy and then subsequently find the partitions that give the best performance.

5.3.1 Cost metric for partitions

Our servicing policy is to allocate a robot to each partition. Subsequently, each robot is responsible for servicing demands that arrive to vertices inside their partition. If there are no demands to service in partition i, then the robot i should wait at the point that minimises the travel distance to each pick location in its partition. This can be found by computing the centroid of the set P_i. If there are demands to service in the queue of the robot, the robot waits until a batch of size R_c of demands have arrived. These batches are then added to a queue and served according to a FIFO discipline. To service a batch, the robot computes a shortest tour consisting of the demands in the batch and the depot and then follows the tour to service the demands. Using standard Kendall notation [159], this system is classified as a G/G/1 queue due to a general arrival time distribution, a general service time distribution, and each queue is serviced by a single robot.

We denote the arrival rate in the set P_i as λ_i, where $\lambda_i = \sum_{\mathbf{x}_u \in P_i} \lambda_u$. Recall that the Poisson process is defined by the Poisson distribution. The Erlang distribution is complementary to the

Poisson distribution, and it counts the amount of time until the occurrence of a fixed number of events. Hence, observe that the inter-arrival times for batches of size R_c are randomly distributed according to an Erlang distribution with mean R_c/λ_i and variance R_c/λ_i^2. Let $\mathbb{E}[L_i]$ be the expected tour length to service a batch of size R_c in partition i and R_cR_p be the total time in each tour for scanning and picking-up the items. Then the expected time to service a demand arriving in partition i is

$$T_i = \frac{1}{2}R_c/\lambda_i + \frac{1}{2}(\mathbb{E}[L_i] + R_cR_p) + Q_i, \tag{5.3}$$

where Q_i is the expected wait time in the queue of robot i. The queue of each robot is a G/G/1 queue with arrival rate λ_i/R_c. In the heavy load regime where the batches arrive frequently, the time to form a batch in each robot's queue is negligible compared to the expected time to service the demands in the queue, i.e. servicing time dominates waiting time. Now, let c_a^2 denote the variance of arrival times to a queue of a robot, and c_s^2 denote the variance of service times. Then an approximation for the expected wait time of a batch in a G/G/1 queue is provided by Kingman's formula [252], i.e.,

$$Q_i \approx \frac{\lambda_i(\mathbb{E}[L_i] + R_cR_p)}{R_c - \lambda_i(\mathbb{E}[L_i] + R_cR_p)} \left(\frac{R_c/\lambda_i^2 + c_s^2}{2}\right)(\mathbb{E}[L_i] + R_cR_p). \tag{5.4}$$

Note that fulfilling an order may consist of picking from different sets in the partition. Therefore, for any given partition $P_1, \ldots, P_m$ of the environment, the number of items picked by robot i with the highest waiting time Q_i is a bottleneck on the total number of orders fulfilled in the warehouse. Therefore, the goal is to find a partition of the environment which minimizes the maximum expected wait time of the demands across all partitions, i.e.,

$$\min_{P_1,\ldots,P_m} \max_{i\in[m]} T_i \approx \min_{P_1,\ldots,P_m} \max_{i\in[m]} \frac{1}{2}\left(\mathbb{E}[L_i] + R_cR_p\right) + Q_i. \tag{5.5}$$

Observe that for a given partition of the warehouse $P_1, \ldots, P_m$, in the heavy load regime, the expected wait time for a demand is a monotonically increasing function of $\lambda_i(\mathbb{E}[L_i] + R_cR_p)$.

Therefore, the problem of minimising the maximum expected wait time is equivalent to

$$\min_{P_1,\ldots,P_m} \max_{i\in[m]} \lambda_i(\mathbb{E}[L_i] + R_c R_p). \tag{5.6}$$

From (5.5), it can be seen that as the size of the set P_i increases, the time to form a batch decreases, and the expected tour lengths increase. This, combined with the desire to operate with as few robots as possible, motivates finding partitions that work near heavy load, i.e., $0.5 \leq \lambda R_p + \frac{2\lambda\bar{r}}{mR_c} \leq 1$ where $\bar{r}$ is the expected distance of items to depot [178]. It is shown that under heavy loading regimes, the unbiased Travelling Salesman Problem (TSP) policy is optimal for servicing demands [46, 253]. This motivates computing partitions that are designed to balance the length of the tour and are equitable to optimise (5.5).

5.3.2 Equitable workload partitioning

Our approach for computing regions of equal workload consists of initialising a set of partitions and then using an iterative local optimisation method to adjust these partitions until they are equitable according to our cost metric.

Recall that the warehouse stores a known set of items $\mathcal{I} = \{I_0, \cdots\}$ and their pickup locations $\mathcal{X}_I = \{\mathbf{x}_0, \cdots\}$ and we want to partition the warehouse and the corresponding pickup locations into P partitions $\mathcal{I}_i \subset \mathcal{I}$

We create partitions by partitioning the warehouse using a generalised Voronoi diagram called a *power diagram*. A power diagram consists of *power cells* that are defined by a generator location and a weight parameter. The cell contains all points that have the lowest *power distance* (weighted distance) between the point and a generator. The power cells are used to compute the partitioning by allocating each point to the power cell that has the lowest power.

More formally, let each partition contain a power cell defined as $C_i = (g_i, w_i)$ where g_i is the generator, and w_i is the weight for partition i. Let ω_i be the centroid of the partition C_i that minimises the distance to each pickup location.

To allocate a pickup location $\mathbf{x}$ to the power cell C_i that describes the partition i, we use a modified version of the *power of a point equation* [248],

$$\text{P}(\mathbf{x}, C_i) = p_{\mathbf{x}}^2 - w_i^2. \tag{5.7}$$

We define $p_{\mathbf{x}}^2$ as the collision-free shortest path cost in the warehouse $\mathcal{W}$ for traversing from the centroid ω_i of power cell C_i to pickup location $\mathbf{x}$.

Calculating the cost of time to service a batch for an arbitrary distribution of items in orders can be analytically difficult or overly specific. Therefore, we use sampling to compute an approximation of the time it takes to service a batch. For a given finite length horizon H, let S be the set of items that are contained in a set of sample orders drawn from a simulation of the spatio-temporal Poisson process $\mathcal{P}$. Using the modified power of a point equation (5.7) the sampled item demands can be allocated to the partitions by the following

$$P_i^s = \{s \in S | \text{P}(s, C_i) < \text{P}(s, C_j), \forall C_j \in C - \{C_i\}\}. \tag{5.8}$$

Using these partitions and allocated samples, we can compute an approximation of the batch service time. The cost of servicing a batch consists of the expected tour length $\mathbb{E}[L_i]$ to service a batch of size R_c, the time it takes to return to the depot, and how long it takes for a batch to form.

To compute this cost, we use approximations: the time for batches to form is approximated as the number of batches formed over the finite horizon H, the depot returns are assumed to be from the centroid of the cell ω_i, and the expected tour lengths are found by computing the expected cost of TSPs over random batches selected from the set of samples allocated to a partition. Formally, the expected tour length is equal to

$$\mathbb{E}[L_i] = \mathbb{E}(\text{TSP}(P_i^s, \omega_i, R_c) + V(\omega_i)). \tag{5.9}$$

Let $TSP(P_i^s, \omega_i, R_c)$ be an approximate polynomial time solution for the TSP that begins and ends at ω_i while making a tour over R_c samples randomly selected from P_i^s. The function V returns the round trip cost of returning to the depot from the power cell's waiting point ω_i. Then using that expected tour length and the other components, the cost to service a region over the horizon is

$$Z_i = \frac{|P_i^s|}{R_c} \left(R_c R_p + \mathbb{E}[L_i] \right). \tag{5.10}$$

5.3.3 Iterative optimisation

This cost is then used in an iterative local optimisation method to adjust the weights of the partitions based on their cost relative to their neighbours to produce the set of equitable partitions with respect to (5.10). Our partitioning algorithm produces a fixed number of partitions, and we allocate one robot per partition, so it is initialised by creating a partition for each of the m robots. With no loss of generality, we chose to initialise the partitions as approximately spatially equal since, in practice, it works well. The initial partitions are found using the Lloyds algorithm with the kmeans++ seeding algorithm to create the initial partitions P that are approximately equal in size. A power cell is created for each partition and is initialised with equal weight. These initial weights are the starting point for optimisation.

The initial partitions are roughly equitably sized. However, they are not balanced according to the expected cost of servicing a partition (5.10). To balance the costs of servicing these partitions, we iteratively optimise neighbourhoods by adjusting their relative power cell weights. To perform the optimisation, let W be a tuple containing the weight of each power cell $W = (w_1, \dots, w_m)$. Then, the cost of servicing all the regions is

$$H_v(W) = \sum_{i=1}^{m} w_i.$$

The difference in workload between partitions is computed as

$$RC_i = \sum_{j \in N_i} \frac{1}{2\gamma_{ij}} \left(\frac{1}{w_j} - \frac{1}{w_i} \right) / H_v(W),$$

where $\gamma_{ij} = ||g_j - g_i||$. N_i is the set of neighbouring partitions for partition i. This local difference is used to adjust the weights according to

$$w_i' = RC_i \cdot \alpha + w_i$$

where α is a scalar selected to control the step size of each iteration. The new weights are used to update the weight tuple W that defines the new iteration of power cells:

$$W = (w'_1, \dots, w'_m).$$

The set of weights is updated simultaneously then the new partition allocations are computed. This method results in changing the weights by their portion of the total cost and their cost relative to their neighbours. This results in partitions that are more expensive than their neighbours and make up more of the total cost getting a negative weight which shrinks their partition size. Similarly, lower-cost partitions get a positive weight and increase their partition size.

This process continues until the allotted calculation time elapses or the system stabilises on a local minimum.

5.4 Analysis

The overall problem is computationally hard, mainly due to multiple agents and order uncertainty in space and time. In order to address the inherent challenge, we presented a number of heuristics and approximations. For example, we randomly sample from the process $\mathcal{P}$ rather than finding an analytical solution to simulate demand. Therefore, our sampling-based approach applies to any warehouse setting with complex and arbitrary representations of order estimates. On top of that, this approach is known to work well in practice [105].

Solving a TSP optimally is NP-Hard, so using it as a part of the partitioning cost seems detrimental. However, for conventional rectangle warehouse layouts, an optimal TSP can be computed in linear time [254]. For more generic warehouses, there still exist good quality polynomial-time approximations [255].

Since the partitioning algorithm produces valid partitions each iteration, it is *anytime* and can be stopped for the current best solution or left to run for iteratively improved solutions. This is possible because the computational complexity for computing an iteration is polynomial and can be run for a finite number of iterations, all of which give a solution.

Our method scales with the problem size and number of robots due to computing the optimisation over local neighbourhoods; it only needs to consider a subset of neighbouring partitions for each partition instead of all pairwise combinations. In addition, we avoid the batching and allocation problems with our policy choice. Online, each robot only needs to consider jobs inside their partition and follow the policy to service them. The decomposition into a series of single robot problems avoids needing to plan in the joint space.

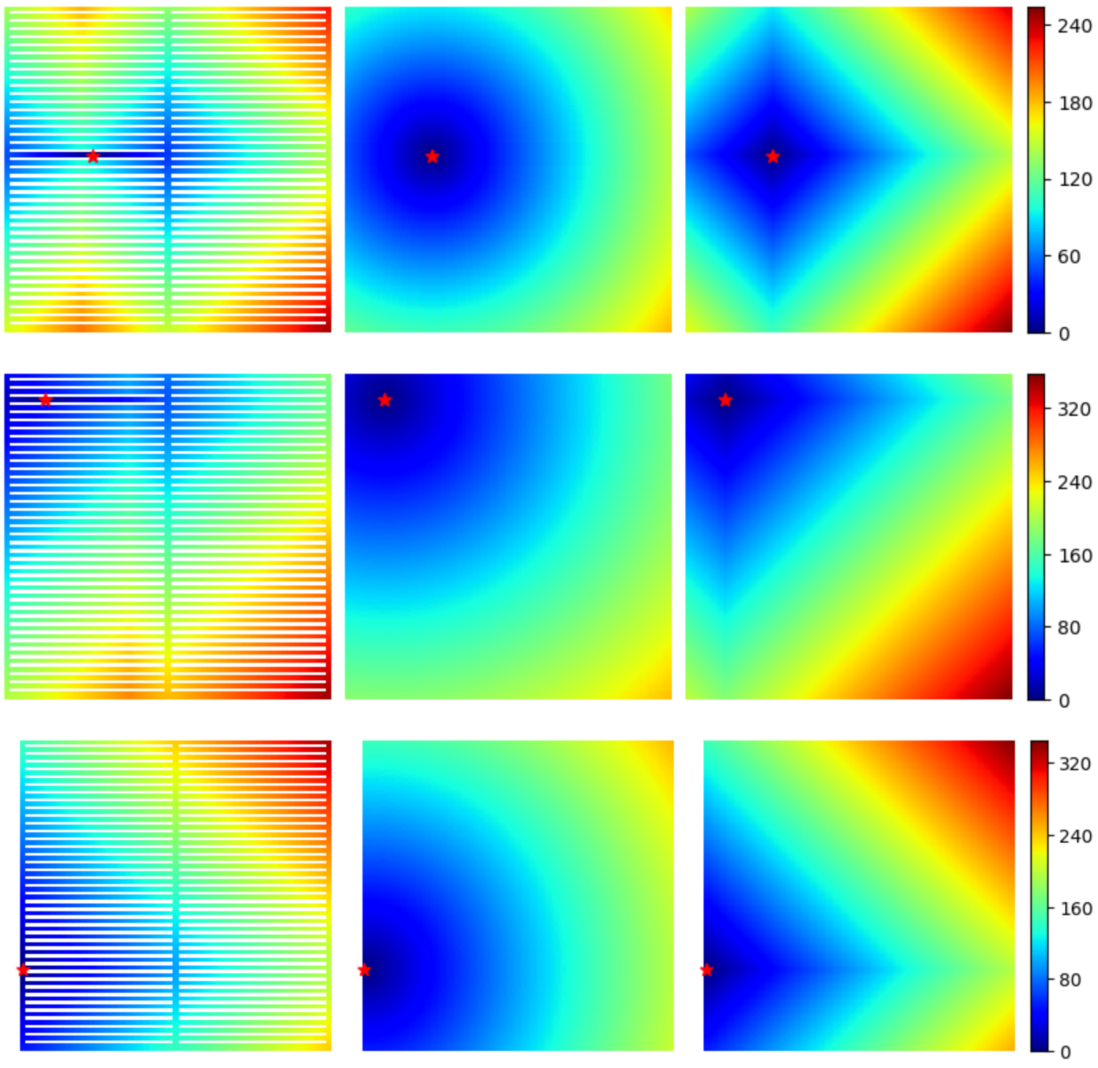


Figure 5.2: Figure showing the change in the distance around the warehouse according to different cost metrics. The distance is shown as a heat map originating from the red star to all points in the warehouse. Column 1 is our distance metric, column 2 is L2 norm (Euclidian), column 3 is L1 norm (manhattan distance). Both L1 and L2 norm are computing without obstacles to demonstrate that the distance considering the obstacles is different to a distance metric that respects the additional cost of routing around shelves.

Computing the TSP around the warehouse and the shortest path between points needs to be computed according to a cost metric. We show some examples of different cost metrics in Figure 5.2; the leftmost column is our cost metric, the middle column is the L2 norm (Euclidian), and the rightmost column 3 is the L1 norm (manhattan distance). From these plots, it can be seen that the cost of traversing around shelves is almost like the L2 norm, but it changes based on the location of the agent inside the shelves. Changing the metric used changes the cost between points which in turn can change the partitions that are created. This phenomenon is explored further in the partitioning results Section 5.5.3.

5.5 Simulation

This section outlines the setup for the simulation environment that we use to validate our algorithms' performance and use these empirical results to validate our algorithm. We also provide some insight into the effect of various parameters on the performance of our method vs a typical *order-picking* process or the previous equitable partitioning method. Finally, we conclude with an analysis demonstrating our method's effectiveness.

5.5.1 Simulation setup

Our simulation environment consists of 5-robot (*pickers*) operating in a two hundred-meter by two hundred-meter picking area within a two-block (rows are split into two blocks) warehouse with a two-meter spacing between rows and 1200 pick locations. The geometry of our simulated warehouse is based on a typical fulfilment warehouse owned by a logistics company that does manual-picker-to-part operations [199]. The parameters we selected for our simulations are similar to those used in multiple previous works addressing the dynamic warehouse order-picking problem. Additionally, our work does not consider the problem of selecting between multiple locations for the same item or the storage capacity of a specific item at a location, as these are not concerns for certain classes of warehousing operations [138, 194, 198, 256].

The spatio-temporal compound Poisson process $\{\mathcal{P}(t)\}_{t\in\mathbb{R}+}$ is defined time-intensity parameter λ. For the simulations in this work, we model the item-quantity $\psi \sim Binomial(n, p)$ as a binomial distribution with a sample size n of 5 and a probability of 0.625 which is roughly equivalent to a normal distribution with a mean of 5 and variance of 2. The item spatial order density function ϕ determines the likelihood for each item to be included in the order is modelled as a discrete uniform distribution ϕ. We compare how the tested methods perform under heavy loading $0.5 \leq \rho \leq 1$.

The robots in our simulation have a varying amount of capacity R_p depending on the simulation. The robot travels at a constant walking speed of 1 meter per second. We assume it takes a constant five seconds per item to pickup R_p and drop-off R_d. In this work, to find the warehouse traversal road map, we use the Fast Marching Tree(FMT) algorithm [257] and build this map offline. For this work, we assume that all items are similarly sized and, hence, have a capacity cost of 1. Additionally, we ignore changes in speed caused by moving carts around corners and robots blocking rows with their carts. Finally, we assume that all robots are homogeneous. The hardware used to perform the experimental evaluation is a desktop computer with a 17-6700 CPU and 32Gb system memory, running a 64-bit Ubuntu 15.10 operating system. The software was written in 64-bit python 3.6 The experimental results are averages of 60 simulation trials (approximately two months of operation) that are 10-hours long in the same environment with different sets of orders drawn from the Poisson process for each day. Each algorithm is tested for each day using the same set of demands drawn from the Poisson process to ensure that the only factor changing is the algorithm the robot team uses to service the dynamically arriving unknown demands.

The equitable partitioning algorithm we compare against is from [46] where the partitions are equitable with respect to ϕ, but it has no considerations for the obstacles by using Euclidean distance rather than the obstacle-free route distance, and it does not account for depot returns. However, once these partitions are found, they follow the same servicing policy as our algorithm, so the main point of comparison is how the partitions are found. To identify what features of our method are most beneficial and gain some comparative insight, we compare our method against two methods found in practice: single-order with optimal routing and equitable partitioning without modification. Single-order picking is the policy where whole orders are naively allocated to a robot without consideration for future orders or sharing work between robots. However, it is still practised by industry due to its simplicity. For the single-order method, a robot is allocated a batch of items, and it follows a near-optimal route found by a TSP solver and visits the items on their list, then returns to the depot. By comparing these two methods, we can understand if the partition-based policy or the balancing of partitions alone has a crucial role in the performance of the system and demonstrate the importance of selecting an appropriate cost metric.

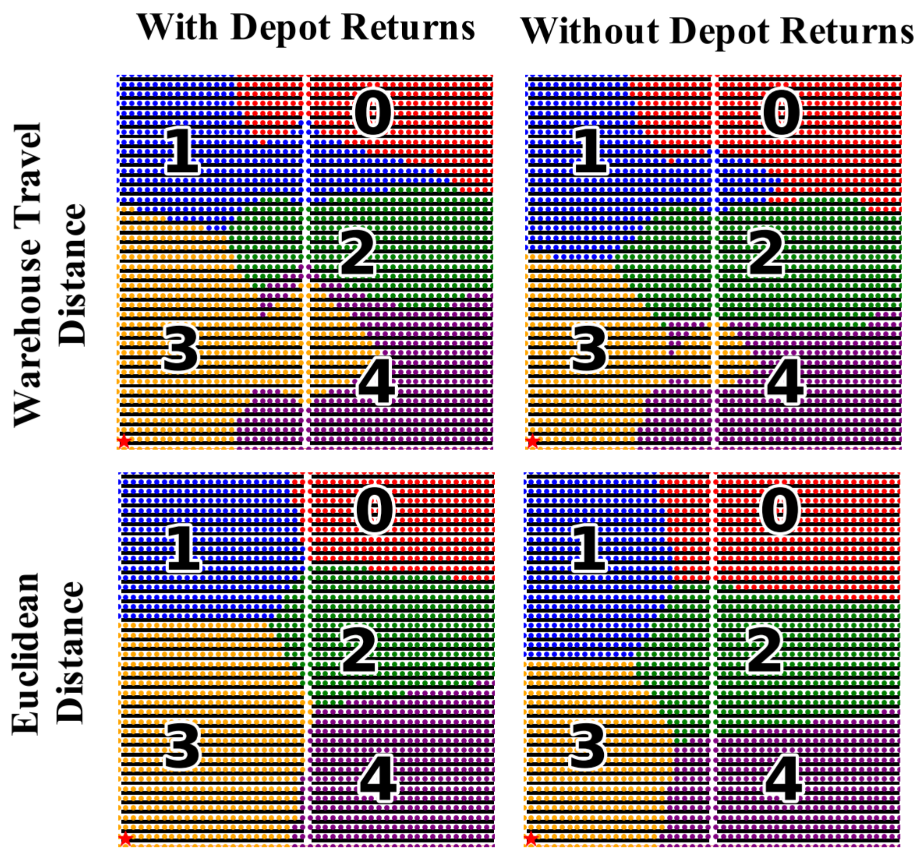
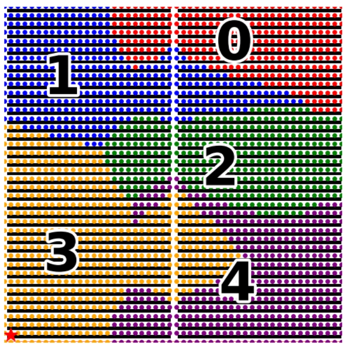


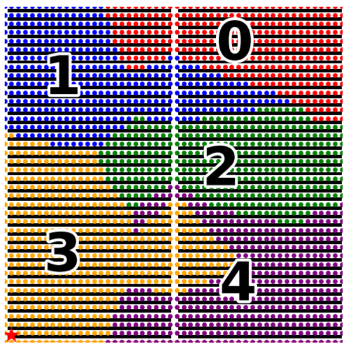
Figure 5.3: Partitions created for five robots with return depot D (red asterisk) showing how our method (upper left corner) differs from existing methods. The results show that considering depot return in the cost metric in (5.10) induces larger partitions near the depot. Similarly, the layout of the warehouse affects the partition since it affects the practical travel distance.

5.5.2 Cost metric component effect

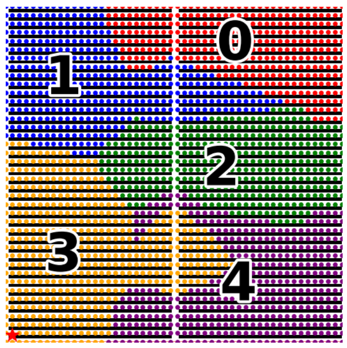
To examine the effect of different cost metric components on the partitions created, we note that (5.10) consists of two main components of cost. First, the expected tour length in the warehouse graph to service demands, which we call *TSP* and the cost to return to the depot to drop off the collected items, we call *depot return*. We disable the TSP cost component by replacing it with obstacle-free Euclidean distance when computing the expected tour lengths and setting the depot return costs to 0. In Figure 5.3, we show the different partitions that are generated when different parts of the cost are changed. Disabling the TSP cost component produces partitions shaped according to Euclidean distance (ignoring obstacles) to the power generator instead of using the routes in the warehouse. Turning off the depot returns means that partitions are sized evenly throughout the warehouse. When depot returns are used, the partitions are larger when closer to the depot than further away, which implies that the closer-to-depot partitions can cover more item locations for the same amount of work. This test helps to explain the performance differences between our method and the equitable partitions produced by the Bullo method. Their partitions are equivalent to the disabled TSP and disabled depot return.



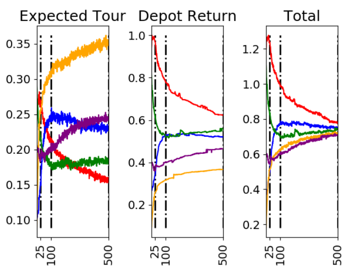
(a) $R_p = 1$



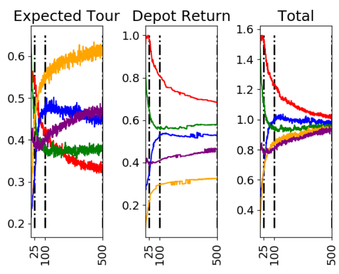
(b) $R_p = 3$



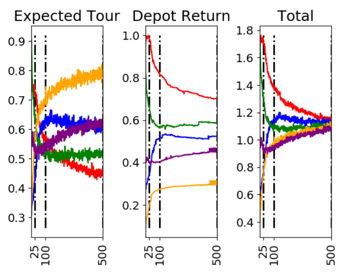
(c) $R_p = 5$



(d) $R_p = 1$



(e) $R_p = 3$

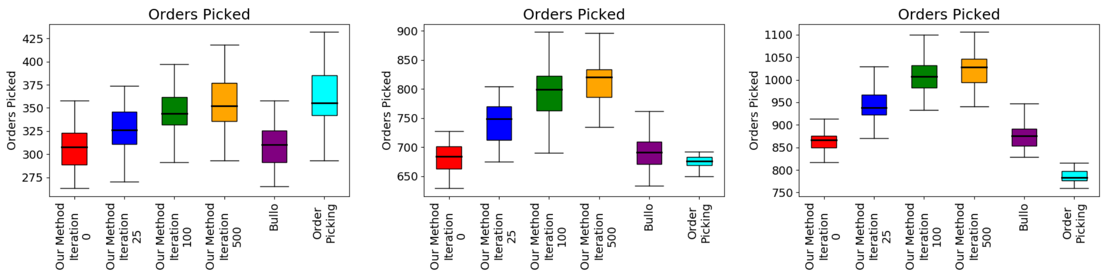


(f) $R_p = 5$

Figure 5.4: (a, b, c) Equitable partitions produced by our method for five robots with capacities $R_p = \{1, 2, 3\}$; capacity affects the partitions as capacity increases the importance of depot returns decreases. Partition 3 (in yellow) becomes smaller at higher capacity, while Partition 0 becomes larger. (d,e,f) Normalised costs of each partition over iterations. Tour costs for a single partition may increase to reduce the depot return costs for another partition, resulting in lower overall totals.

5.5.3 Partitioning balance results

To examine the quality of our *equal work* partitioning, we tested a scenario with five robots and allowed the algorithm to run, then observed the costs for each partition after each iteration. The results of 500 iterations of the algorithm for various capacities are shown in Figure 5.4. The first row shows that the capacity of the robots changes the partitions that the algorithm produces; as capacity increases, the importance of depot returns decreases. This effect can be seen by observing



(a) $R_p = 1$ (b) $R_p = 3$ (c) $R_p = 5$

Figure 5.5: Performance comparison between our method at different iterations and Bullo [46] and order picking for five robots with varying capacity $R_p = \{1, 2, 3\}$, where order picking is a baseline greedy method without partitions. The improvement as we increase the capacity R_P is shown in (a, b, c).

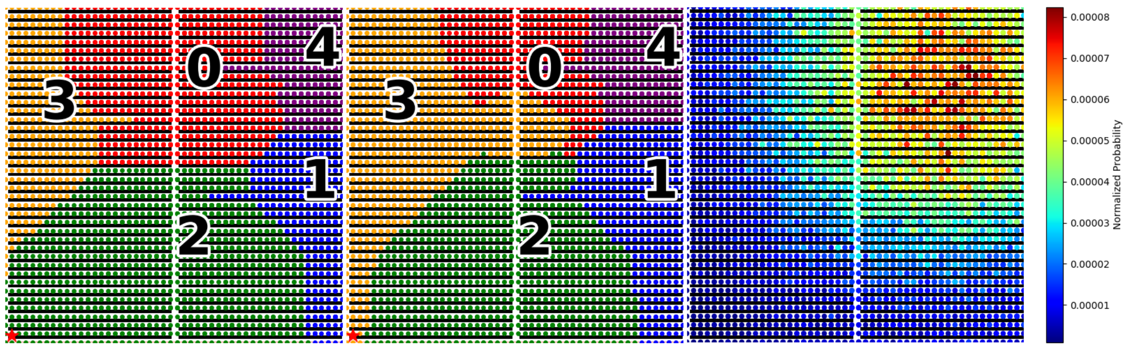
that partition labelled three becomes smaller at a higher capacity, and similarly, partition 0 becomes slightly larger.

The second row shows that initially, the amount of work needed to service each partition is quite unbalanced, with the most expensive partition taking nearly six times the amount of work to serve as the least expensive according to our cost metric. After 500 rounds, our algorithm reduces the gap to within 5-10%; these results are consistent with most of our tested scenarios. Our algorithm produces near-balanced results that, when used in practice, can produce high-quality improvements over the unbalanced initial partition configuration. This result demonstrates that the gradient descent balances the workload between partitions and that the anytime aspect of the algorithm is useful. Additionally, It can be seen that tour costs for a single partition may increase to reduce the depot return costs for another partition, which results in overall lower totals. Recall that increasing the partition size increases the frequency of batch forming and the expected tour cost. Therefore, growing partition three and shrinking partition 0 allow the system to increase the frequency that batches form in 3 and reduce how often partition 0 has to return to the depot, which is comparatively more expensive than returning from partition 3. This result shows that our method can balance the costs between partitions through the growing and shrinking of the size of neighbouring partitions.

5.5.4 Order density partitioning results

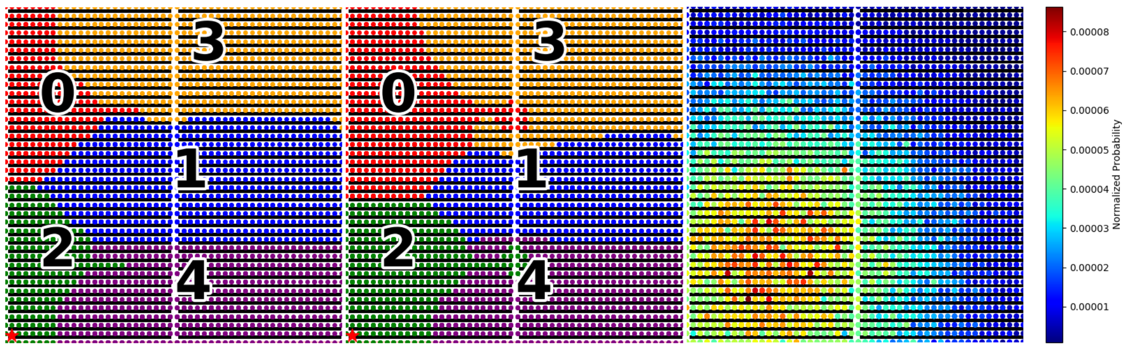
To illustrate how the partitions' shapes can vary with order density, we present partitioning results under different order density distributions. We model the item spatial order density function ϕ, as a truncated discredited normal distribution. This allows us to explore orders containing items predominantly localised in a specific area. We examine two scenarios where items in orders are mostly selected from either the bottom left of the warehouse (near the depot) or the top right (far from the depot). We center the distribution at $\mu_x = 50, \mu_y = 50$ and $\mu_x = 150, \mu_y = 150$. To ensure the distribution spares the entire warehouse, the standard deviation needs to be large enough so that the entire workspace is within three standard deviations to allow each item a sufficient chance of being ordered. Using the selected distribution centres, the needed standard deviation can be computed using $\sigma = \sqrt{(\sigma_x - 0)^2 + (\sigma_y - 0)^2}/3 = \sqrt{(150-0)^2 + (150-0)^2}/3 \approx 73$ so the standard deviation needs to be larger than 73. Hence, we set $\sigma_x = \sigma_y = 75$.

Using the aforementioned distributions, we present the partitioning results for the bottom-left distribution in Figure 5.7 and the top-right distribution in Figure 5.6. It can be seen that the partitions created using these distributions vary and are differently shaped from those created using a uniform distribution, such as those in Figure 5.4. The initial state from the weighted Voronoi reflects the underlying distribution, but our optimisation process refines this initial partitioning to include the impact of travel time. Notable, even with non-uniform distributions, the optimisation shrinks partitions further from the depot and expands those closer, demonstrating the influence of the travel time component.



(a) Initial state (b) After 500 iterations (c) Order density distribution

Figure 5.6: Partitions created for 5 robots with the spatial order distribution defined as a Gaussian distribution centered in the top right of the warehouse far from the depot.



(a) Initial state (b) After 500 iterations (c) Order density distribution

Figure 5.7: Partitions created for 5 robots with the spatial order distribution defined as a Gaussian distribution centered in the bottom left of the warehouse near the depot.

5.5.5 Orders picked results

These box plots were created from sixty runs of a 10-hour workday simulation for 5-robots with varying capacities. In addition, we tested our servicing policy on the partitions generated by our gradient descent algorithm at several iterations to demonstrate that with more iterations, performance should improve if the metric being balanced is applicable.

We compare against the equitable partitions from literature called ‘Bullo’ here. The baseline comparison *order-picking* is not a partitioning-based method; it greedily allocates an order to an idle robot and adds it to the queue to allocate later when a robot becomes free.

Figure 5.5(a) shows the results for robots with one capacity. In this configuration, our method performs similarly to order-picking as the robots constantly traverse back and forth from their partitions to the depot. These constant depot returns reduce the opportunity for improvement by minimising the travel distance to subsequent items inside a batch. Figure 5.5(b) shows the results for robots with three-capacity. In this scenario, the benefit of having robots distributed around the warehouse becomes apparent as they can reduce wasted travel by being closer to demands when they arrive. Additionally, even the less specific ‘Bullo’ partitioning outperforms greedy allocation. Lastly, Figure 5.5(c) shows that these results continue to hold as you increase capacity. Our method can pick up 10-30% more orders over a simulated workday in the higher capacity cases.

5.6 Summary

This chapter presented a framework for the dynamic warehouse order-picking problem. We demonstrated how modelling as a generic vehicle routing problem limits the scalability and how the type and method of replanning need to be carefully considered. Subsequently, we defined the problem by modelling the warehouse and the arrival of orders to pick as a Poisson process that generates items from an arbitrary distribution. To solve the servicing of the dynamically arriving demands, we need to find a policy that maximises throughput. To find the policy, we developed a computationally efficient partitioning-based algorithm that creates regions of *equal work* suitable for near heavy-load regimes. Using a simple service policy, a robot is then allocated to each partition to service dynamically arriving jobs. We have shown analytically that the method is near optimal for heavy loading regimes. The important contribution of our method is that we address important aspects such as travel time to the depot and non-Euclidean space, which is often neglected. We validated the approach in simulations over a two-month operation under varying capacity constraints; the proposed framework is proven to outperform typical single-order or previous equitable partition methods under heavy loading conditions.

This chapter presented a solution to produce schedules for problems with stochastic tasks. The dynamic warehouse-order picking requires optimising throughput for a scenario where the tasks arrive dynamically throughout the workday. This uncertainty was dealt with by modelling a stochastic process that describes the orders and then creating a policy (instead of a specific plan) that performs well given the operational reality described by that process. Maximising the system's throughput described by that process means that our method accounts for many potential sequences of tasks rather than planning over what is available now. The key idea is to define the policy so that all agents are working equally and that agents are near where work is expected to arrive, thus utilising the team effectively and reducing the cost of completing the work.

In the next chapter, we examine the final component of our probabilistic scheduling framework, scheduling with stochastic resources. In these problems, the resources consumed by completing tasks are replenished through a stochastic process. These problems differ from stochastic task problems because the tasks are fully known, but the plan can become infeasible due to insufficient resources when trying to complete a task. Initially, it seems like stochastic constraint and stochastic resource problems are the same. However, stochastic resource problems differ from stochastic constraint problems, as the process that causes schedules to become infeasible is coupled with the resources consumed through task execution rather than the constraints. Resources are accumulated over time and can be spent to service tasks, so the ordering and timing of tasks are paramount. Whereas in constraint problems, there is no accumulation, and the plans need to deal with the changing constraints.

Chapter 6

Stochastic Resources

In this chapter, we address scheduling problems that have stochastic resources. In a subset of scheduling problems, resources are required to complete tasks and are consumed during the work. The resources are considered stochastic when a stochastic process replenishes them, thus making it uncertain if enough resources will be available when completing a task in the future. The process of obtaining more resources can be uncertain for various reasons, such as unclear production amounts or arrival times.

This fourth component of our probabilistic scheduling framework differs from the other three types of stochasticity presented so far in this thesis. The objective is deterministic, and the tasks are fully known, but the plan can become infeasible due to insufficient resources when trying to complete a task. This makes it most similar to stochastic constraints, but the key difference lies in how constraints and resources interact with the planning. The consumption of resources is based on the planned actions and replenished by a stochastic process. In contrast, constraints change over time in response to the change in the operating environment regardless of the actions taken.

We explore stochastic scheduling problems by introducing the risk-bounded stochastic resource scheduling problem. This problem is a variant of a scheduling problem in which a stochastic process replenishes the resources consumed by jobs, and the produced schedules have a bounded acceptable risk. Our method is based on taking high-quality forecasts of the resource production and then perturbing them into a set of alternative forecasts (with associated probabilities) using the expected forecast precision. This set of forecasts and their probability of occurring is then used to plan a schedule with a bounded amount of risk. This method is applied to a real-world problem of scheduling operations for a hydrogen research facility that consumes the resources produced by uncertain wind power. Finally, we empirically evaluate our approach through extensive backtests.

6.1 Overview

In this chapter, we consider the problem of scheduling a set of jobs with a mixture of typical deterministic scheduling constraints and a stochastic resource constraint. The deterministic constraints we consider are the time bounds of the jobs, the amount of resources consumed by a job, and the number of jobs that can be processed in parallel. A stochastic resource is a replenishable resource such as energy or fuel produced through a stochastic process. Additionally, we are interested in creating schedules that have a bounded risk of failure when executed. A scheduling failure occurs when the produced schedule exhausts its supply of resources before all scheduled jobs have been completed.

The key challenge of this class of problems is that it can be difficult to model the stochastic process that produces the replenishable resource. It would seem possible to estimate the expected amount of resources available at each point in time and then find a schedule that maximises the reward using that expectation. However, it is not sufficient to summarise the state of the system as the expected amount of resources because many sequences of production can result in the same amount of resources at the given time with differing amounts of resources available at previous times. To validate that the schedule stays valid throughout the entire execution of the schedule, a distribution of the resource must be estimated. Additionally, this scheduling problem is non-Markovian due to the dependence of the amount of resources on the history of previous resource production and consumption.

In this work, we explore a real-world problem of scheduling operations for a facility that relies on hydrogen produced by wind power. Wind power is inherently uncertain and difficult to predict. Additionally, for operational reasons, we want to minimise the risk of cancelling jobs that have been committed to, and the facility needs to commit to schedules a week ahead. This future commitment adds complexity to the scheduling problem because forecasting weather over longer horizons becomes more difficult due to increased complexity and removes the possibility of reacting to changing system states.

This problem resembles the renewable energy planning and stochastic resource problems we explored in Section 2.2. However, none of the existing methods are entirely suitable for this problem directly. Existing risk-based methods such as conditional value-at-risk [258] and robust optimisation [259] require modelling the expected distribution of rewards to quantify risk. These methods tend to have strong assumptions such as autocorrelation [260] or being able to model as simple Gaussian or Weibull) statistical distributions [151], which are not practical to apply on the week ahead wind forecast. Wind-powered stochastic resource production is complex, so it cannot be sufficiently modelled using a simple distribution. While data-driven approaches that fit synthetic distributions and historical data offer an alternative, they face their limitations when trying to model a non-stationary, highly non-linear process like wind accurately. Existing methods are insufficient to meet these constraints, and most of the existing literature focuses on using longer-term predictions for strategic decisions [261], which do not require the accuracy and sampling rate predictions needed here. Moreover, typically, they plan a single day [153–155, 262, 263] or rely on recourse to adjust their plans if the schedule becomes infeasible [152]; the problem defined here does not allow these changes due to the commitment requirement. The ideal method must handle medium-term (one week) forecasting at a fine-grained sampling rate (hours) and accommodate the operation constraints of green hydrogen production. To address these challenges, we propose a solution that decouples the forecasting and scheduling components.

Instead of trying to model and predict wind ourselves, we use a forecast from a different model and then model the expected error between the forecast we are using and the real wind. We treat the forecasting data as a prior estimate of the wind, which acknowledges that modern forecasting methods are reasonably accurate but not infallible, so we use them with the expectation that they will have some errors. This consideration of error allows the trade-off between optimality and operational risk to be considered. Assuming more errors in the schedule should result in more conservative resource usage. At the same time, higher levels of trust will give a greater reward and a higher risk of failure if the forecasts are incorrect. This allows us to leverage any suitable forecaster that delivers a good quality forecast with sufficiently dense predictions, allowing us to use complex forecasting models and produce schedules with the constraints and robustness required.

We propose a method to address the risk-bounded stochastic resource scheduling problem that works by taking a forecast of the upcoming time horizon and applying perturbations to produce multiple scenarios with an associated likelihood of each. To minimise risk, we use the forecast as an upper bound, and the perturbations create alternative forecasts that predict lower amounts of production. This one-sided bounding avoids the failure cases that occur due to schedules over-allocating resources. Subsequently, we estimate the risk of a schedule as the total probability of the forecasts that fail. To find a schedule, these scenarios, their likelihoods and the set of jobs are passed into a sampling-based planner, which generates feasible schedules that fail in an acceptable number of cases.

Formally, we create the perturbed forecast sequences and their associated likelihoods by enumerating a Bernoulli process to modify the original forecast and then produce a schedule using a Monte Carlo Tree Search (MCTS). To create the perturbed forecasts, we discretise the original forecast into a series of binary variables over a fixed period of windows representing whether resource production occurs. We estimate the expected precision of the forecast and use this to generate biased sequences. To create the perturbation sequences, we enumerate a binary tree to represent all possible outcomes of the Bernoulli process with success probability related to the expected precision of the forecast. These sequences are then used to mask out the positive predictions of the forecast. This works because it assumes errors come in blocks related to discretisation window size. It assigns a higher probability to sequences close to the forecast and less to ones farther away. Sequences and their probabilities are then used in a Monte Carlo Tree Search (MCTS) algorithm to produce valid schedules that meet the constraints. A benefit of this algorithm is that it has the *anytime* property and has a fast computation speed, so it can be used to evaluate various parameterisations of the problem.

Using this method, we can produce valid near-optimal schedules quickly with an acceptable level of risk/robustness. Our approach to solving this problem offers several vital contributions over the existing methods: we formally define the risk-constrained stochastic scheduling problem, introduce a new method to produce scenarios with associated probabilities, demonstrate a high-quality scheduler that is able to meet the specified constraints, and finally demonstrated our approaches ability to produce near-optimal schedules on a real-world dataset.

6.1.1 Chapter outline

The remainder is this chapter is organised as follows. Section 6.2 formally defines the resource scheduling problem that would be solved if we had perfect information. Subsequently, it defines the stochastic resource variant with additional risk constraints. Section 6.3 describes our process to solve the stochastic resource risk-constrained scheduling problem. It first outlines our process of creating the perturbed forecasts and their likelihoods that are used in the scheduler as stand-ins for the stochastic process. Next, we explain our sampling-based scheduler algorithm that produces the schedules. Next, Section 6.4 describes the real-world green hydrogen research scheduling problem and presents simulation experiments that empirically prove the effectiveness of our proposed method. Finally, Section 6.5 summarises the chapter.

6.2 Problem formulation

We consider the problem of scheduling jobs that consume a resource produced from some process with an uncertain outcome, which subsequently makes the available amount of resource stochastic. This uncertain amount of resources requires consideration of the likelihood of having a given amount of resources available at any time. We are interested in maximising the utility of completing jobs while limiting the risk of a schedule that requires more resources than is available. We define the problem in more detail in the following section.

Given a set $\mathbf{J} = (J_1, J_2, \ldots, J_n)$ of n jobs to schedule in a set $\mathbf{L} = (L_1, L_2 \ldots, L_m)$ of m possible homogeneous locations that allow m jobs to be completed in parallel. Each job $J_i \in \mathbf{J}$ has an associated resource expense $E_i \in \mathbf{E}$ that is consumed during the job's execution, and an associated reward or utility $U_i \in \mathbf{U}$ that is gained by completing this job. Let $\mathbf{T}$ be the finite planning horizon $(1, 2, \ldots, T)$ over which the schedule will take place with a total length T.

A schedule consists of a sequence of decisions that represent the allocation of a job to a location at a specific point in time. Let the set of all possible time, job, and location decisions be $\mathcal{X}$ and from this, define the set of selected decisions that make up a schedule to be $\mathbf{X} \subset \mathcal{X}$. Subsequently, $x_{i,l}^{t} \in \mathbf{X}$ means that job J_i is scheduled to start at time t at location L_l. Because the locations are homogeneous, tracking the allocated locations is unnecessary as long as the method ensures that fewer than m jobs are scheduled simultaneously. Therefore, without loss of generality, for simplicity of the notation, we can define $x_i^t \in \mathbf{X}$.

The system has an initial resource budget of B_0, and for problems where the budget represents a physical property such as fuel, stored energy, or inventory, there is a finite maximum budget of B_{max}. The additional resource is not stored if the additional production exceeds the maximum capacity. Given a sequence of expected production of the resource over time $\mathbf{C} = (C_1, C_2, C_t, ..., C_T)$, where C_t is the amount of production at time t. Similarly, let the total amount of resources used by a schedule from time $t = 0$ to time t be $\phi(\mathbf{X}, t)$. Using these definitions, the capacity at time t can be calculated as follows:

$$B(\mathbf{C}, \mathbf{X}, t) = \begin{cases} B_0 + C_t - \phi(\mathbf{X}, t) & \text{if } B_0 + C_t - \phi(\mathbf{X}, t) < B_{max} \\ B_{max} & \text{if } B_0 + C_t - \phi(\mathbf{X}, t) \geq B_{max}. \end{cases} \tag{6.1}$$

For this problem, we want to bound the amount of risk, so let λ be the acceptable level of risk for the selected schedule. Where risk is defined as the likelihood that a schedule will fail. Schedule failure occurs if at any time the resource budget is negative $B_t < 0$. Subsequently, we need to estimate the probability of failure given a schedule and a model of resource production. Let $\mathrm{P}(\mathbf{C}, \mathbf{X})$ be the probability of failure of a given schedule. Therefore, a schedule is valid if $\mathrm{P} \leq \lambda$.

We formally define the risk-bounded stochastic resource scheduling problem as follows:

Problem 6.1 (Risk-Bounded Stochastic Resource Scheduling Problem)**.** *Given set of n jobs $\mathbf{J}$ with associated utilities U and costs E, the expected production of the resource $\mathbf{C}$, find a set of optimal job, time, location allocation decisions $\mathbf{X} \in \mathcal{X}$ over time horizon T that maximises their utility U_j, not exceeding the resource budget $B_t > 0 \forall t$ and ensuring the risk is below the threshold*

$\mathrm{P}(\mathbf{C}, \mathbf{X}, t) \leq \lambda$.

$$\begin{aligned}
\max_{\mathbf{X}\in\mathcal{X}} \quad & \sum_{i=0}^{n}\sum_{t=0}^{T} x_j^t.U_j && \forall x \in \mathbf{X} && \text{(6.2a)}\\
\textit{subject to} \quad & \sum_{t=0}^{T} x_i^t \leq 1 && \forall i \in (1,\ldots,n), && \text{(6.2b)}\\
& \sum_{i=0}^{n} x_i^t \leq m && \forall t \in (1,\ldots,T), && \text{(6.2c)}\\
& B(\mathbf{C}, \mathbf{X}, t) \geq 0 && \forall t \in (1,\ldots,T), && \text{(6.2d)}\\
& \mathrm{P}(\mathbf{C}, \mathbf{X}, t) \leq \lambda, && \forall t \in (1,\ldots,T), && \text{(6.2e)}\\
& x_j^t \in 0, 1. && && \text{(6.2f)}
\end{aligned}$$

This problem formulation, when solved, gives a solution to the risk-bounded stochastic resource scheduling problem, and the constraints define the rules to which the schedule must adhere. The objective function 6.2a sets the goal of maximising the total utility gained from all selected jobs while adhering to the following constraints. The constraint 6.2b ensures that each job can only be selected once, which limits the possibility of selecting duplicate jobs to gain additional reward. The next constraint 6.2c models the system limitation that no more than m many jobs can be selected at any time, and there are only m positions, so the system cannot service more jobs in parallel. Additionally, the constraint 6.2d forces schedules where the total budget used by jobs, while considering the generated resource, does not exceed the budget at any time; this ensures that the schedule cannot use resources that are not available at the time t. Then we have the constraint 6.2e that ensures that the probability of the schedule failing is less than the acceptable risk level denoted as λ. Finally, constraint 6.2f means that a job selection is discrete, meaning that a job is either fully allocated or not allocated at all; this removes the possibility of partial job selection. The budget 6.2d and risk 6.2e constraints include the stochastic sequence of resource production, so particular care must be taken when selecting the sequence and the calculation of these constraints. The sequence is a time series of production, so there are temporal and magnitude components that both need to be correct for schedules to be properly validated.

6.3 Risk constrained scheduling with stochastic forecasts

In the following section, we introduce our methods for solving the risk-bounded stochastic resource scheduling problem, which in particular addresses the uncertainty of the stochastic resource in a way that considers the probability of the temporal component and the total production of the resource.

The available budget for the system B_t is produced from the stochastic process. If the amount of resource varies from the expected sequence $\mathbf{C}$, then the budget available at time t is different from what is expected, which can cause valid solutions to the scheduling problem (Problem 6.1) to be actually invalid. In this problem, the resource is produced by a sequence of hidden variables, and in practice, only a noisy forecast of what value this variable will take is available. We know that the forecast is noisy, but we do not know the amount of noise or when the error will occur during the sequence. To account for this, we want to turn the singular forecast into a distribution of the resource over time, and then ensure that the schedule is valid given with respect to the probability we assigned to the resource distribution. If the problem can be modelled as a distribution or a stochastic process, then we could use this to get the probability that for various sequences and meet the risk and stochastic budget constraints. However, in this work, we are interested in scheduling using resources generated by some external process that is difficult to model and thus want to use a model-free data-driven method. In particular, the motivating problem for this work generates resources via wind power, and wind is a system that is too complex to be accurately modelled in a closed form. Hence, we propose a method that uses a forecast (such as those generated by complex meteorological models) as a prior, which is then modified to create candidate scenarios with associated probabilities. This process does not require that we have an explicit model and can generate the predictions and probabilities from the data directly.

We propose approximating the resource production distribution by creating a variety of candidate scenarios given the forecast data and assigning the probability that the scenario will occur based on an expectation of the amount of error.

6.3.1 PPV based forecast perturbation

Given a forecast $\mathbf{C}$ of the expected production of resources over time and an associated forecast uncertainty C_σ^T. We want to turn the forecast $\mathbf{C}$ into a discrete finite set of k production sequences with associated probabilities $\mathcal{C}_* = ((\mathbf{C}_1, P_1), (\mathbf{C}_2, P_2) \ldots (\mathbf{C}_k, P_k))$. $\mathbf{C}_i$ is the i-th sequence in the set and P_i is the associated probability that that sequence occurs. This set of sampled sequences and their associated probabilities approximate the distribution of potential futures.

Using this set of sequences it is possible to compute the probability that a schedule will fail. The probability of a schedule failing given the potential production sequences is:

$$\rho(\mathbf{C}_i, \mathbf{X}, t) = \begin{cases} 0 & \text{if } B(\mathbf{C}_i, \mathbf{X}, t) \geq 0, \forall t \in (1, 2, \ldots, T) \\ P_i & \text{if } B(\mathbf{C}_i, \mathbf{X}, t) < 0, \forall t \in (1, 2, \ldots, T). \end{cases} \tag{6.3}$$

This means if a schedules budget is valid for all points in time its failure probability is equal to 0 else it is the probability of that sequence occurring. Similarly, the probability that a schedule is valid is equal to the sum of all probabilities for all production sequences in $\mathcal{C}_*$. The probability of a schedule failing by time t can be calculated using

$$\mathrm{P}(\mathcal{C}_*, \mathbf{X}, t) = \sum_{\mathbf{C}_i \in \mathcal{C}_*} \rho(\mathbf{C}_i, \mathbf{X}, t). \tag{6.4}$$

Using this set of production sequences and probabilities, it is possible to compute the probability of schedule failure and, subsequently, the risk of a schedule failing. The question is, ‘How do you create these sequences and assign them a probability in such a way that makes sense?’. Given that we cannot model the underlying process that generates the wind speed, one naive approach would be to randomly sample from all possible sequences and assign them an equal probability. Alternatively, we have a forecast that we know is of good quality, and we could estimate the possible amount of error and use this to create the sequences. Looking at the type of error, we first note that the direction and type of error are important to consider for this problem. Schedule failures occur when there are too many Type-I or false positive errors between the forecast production sequence and the true production sequence. That is, if the forecast says there is generation at a point when there is really no generation, the schedule allocates more resources than would actually be available. This problem is exacerbated when the error occurs together in blocks due to having less time for Type-II errors (false positives) to cover up the previous forecast error and give the schedule more resources than it expected at a point in time. Assuming that we can estimate the positive predictive value (PPV) of the forecast, meaning that we know that the forecast is going to have some positive error, but we do not know *when* during the sequence the error is going to occur.

Using the expected PPV τ and the original forecast $\mathbf{C}$, we want to compute a set of refill sequences $\mathcal{C}_*$ and their associated probabilities. To define the PPV of the schedule (how much over-allocation we expect our forecast to have), we convert the resource production sequence $\mathbf{C}$ to a sequence of binary values that represent whether there is production or not at each point in time. Let $\mathbf{Y}_{pred}$ be the binarized forecast of production, and let the binarized ground truth refill sequence be $\mathbf{Y}_{true}$. That is to say τ is the expected PPV between $\mathbf{Y}_{pred}$ and $\mathbf{Y}_{true}$.

We build a Bernoulli process binary tree to create the perturbed production sequences $\mathcal{C}_*$. The outcomes of successive Bernoulli trials can be modelled as a binary tree of depth D with its two children representing the success and failure outcomes. The probability of these children is the probability of the root-to-child path. These paths encode a sequence of success or failure trial results according to the trial success probability of ζ. The trial success ζ represents the odds of failing a single Bernoulli trial while the PPV τ is the expected error over the entire sequence. To convert τ to ζ we empirically sample the Cumulative Distribution Function (CDF) for the Binomial distribution. The distribution is parameterised as the number of successes $k = D - 1$, the number of trials $n = D$ and a sample trial likelihood $P \in (0, 1)$. The process is outlined in Algorithm 6.1; the computation repeats with slowly decreasing P until the P that results in the correct distribution is found. This distribution results in the all True trial case aka perfect forecast case, having a probability τ which matches the expected PPV of the forecast and the rest of the probability density is spread over the other trials.

After creating the Bernoulli process binary tree, let $\mathbf{G}$ and $\mathbf{L}$ be all the root-to-leaf paths and their associated probabilities, respectively. To create the perturbed production sequences $\mathbf{C}_*$ we take the original forecast $\mathbf{C}$ and split it into D many windows. This split forecast is then perturbed by using the paths in the Bernoulli process tree to modify the production in each time window and the paths have an associated probability. This process is shown visually in Figure 6.1. To compute the perturbed sequences, for each of the sequences in $\mathbf{G}$ take $\mathbf{C}$ and for each period in D if the step in the sequence is a success, return the forecast as is during that period. If the sequence step

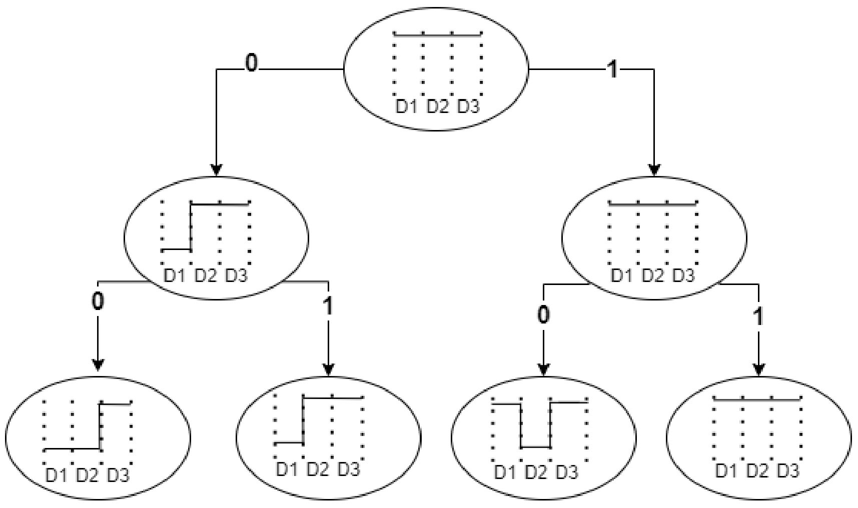


Figure 6.1: Demonstrative figure showing the perturbed forecast generation tree. The root node in the tree is the original forecast. Each child represents a success or failure in a Bernoulli trial. If a trial is a success, the forecast is kept the same. If the trial fails, the forecast is masked out. This process repeats to create all the leaf nodes that represent perturbed forecasts.

is a failure, mask out the production during that period to account for a potential incorrect positive prediction. This process results in the creation of 2^{D-1} many perturbed production sequences. These production sequences and the tree sequence probabilities L are combined to create $\mathcal{C}_*$.

Algorithm 6.1 PPV based binomial trail success finder

```
Require: $D, \tau, \Delta$
Ensure: $P$
  $k = D - 1$
  $n = D$
  $P = 1 + \Delta$
  $B = 0$
  $target = 1 - \tau$
  while $B >= target$ do
      $P = P - \Delta$
      $B = BionomialCDF(k, n, P)$
  end while
  return $P$
```

6.3.1.1 Selection of precision value

The quality of the perturbed forecast probabilities is dependent on the selection of the precision or PPV value τ. If the PPV is set too high the system will assign a higher probability to schedules with little to no error. Similarly, if it is set too low the system will assume a higher amount of error and thus produce schedules that are more conservative. The parameter can be estimated from historical data by comparing historical forecasts against the historical data. In certain problem scenarios such as the problem, we address in this chapter the historical forecasts PPV can be modelled well by the data which results in a principled way to tune the parameter. If this is not possible then the parameter needs to be selected along with the acceptable risk λ based on the practitioners' tolerance for optimality vs safety tradeoff, with larger lambda and lower τ being safer and the opposite being less conservative.

6.3.2 Sampling-based schedule generation

This section outlines our method to produce robust schedules for the stochastic scheduling problem using the perturbed resource production sequences and the expected forecast precision. Given the large search space of the problem, we modify the general Monte Carlo Tree Search algorithm to make it more applicable. The main differences are the integration of the risk to find valid actions in a termination state and the choice of action selection heuristics.

For completeness sake, we include a short description of the Monte Carlo Tree Search algorithm here; for more details, please refer to Section 3.4.1. MCTS is a sampling-based heuristic search algorithm that is suitable for problems with large state spaces, such as scheduling problems, and can deal with complex objective functions and constraints. The algorithm is a biased search that balances exploring possible actions and exploiting previously seen good possible actions to grow the search tree in a manner that results in good solutions faster than random sampling. The algorithm begins with an initial condition (or state) and then iteratively grows a tree representing the search by selecting subsequent actions from this state. If there are multiple possible states, it uses a selection criterion to select a promising state in the tree from which to further explore. From the selected state, successive actions are selected using a sampling heuristic (such as biased random or uniform random), and this process continues until a termination condition has occurred (such as exhausting computation time or resource budget). This action selection process is called a *rollout*. The rollout result is then *back propagated* up the tree from the selected state and updates the summary statistics about these states, which represent the expectation of reward from taking actions from these states.

The MCTS algorithm is used to plan a sequence of allocations that make up a schedule. Given the matrix of perturbed resource production scenarios and their probabilities $\mathbf{C}^*$, the set of jobs, and other parameters of the problem, we want to produce a schedule that meets the constraints of Problem 6.1. We find a valid schedule using a Monte Carlo Tree Search over a tree where the nodes are a (job, location, time) allocation and a path through the tree represents a schedule. The utility function of our scheduler is the summation of the reward for the jobs allocated for a given schedule Eqquation 6.2a. In this problem, we want to maximise the reward gained by allocating jobs using the finite budget. To speed up convergence towards promising solutions, we bias the rollout to more often consider jobs with higher budget-to-reward ratios. Using this ratio, we modify the likelihood of selecting an action. The rollout biasing is computed using:

$$p(x_{i,l}^t|\mathbf{X}) = (E_i/U_i) + \epsilon. \tag{6.5}$$

Where $x_{i,l}^t$ is the action, E_i is the resource expense required, U_i is the utility gained by completing the job, and adding ϵ ensures that all actions are probabilistically feasible to select.

6.3.2.1 Action generation

Due to the combinatorial nature of the action space (every possible joint allocation of job, location, time) and the sparsity of the reward space caused by the large number of invalid schedules, naively considering all actions is not feasible. To speed up convergence to high-quality solutions, we modify the action generation only to consider allocations where the time and slot would be feasible for a candidate job. To achieve this convergence speed-up, we restrict the action space by removing the selection of location and the time allocations, so it becomes a job subset selection problem, and the ordering of the jobs encodes their time and location allocations.

The location selection problem can be avoided (which reduces the size of the action space) by using the fact that for this problem, we assume that the locations are homogeneous. Thus, to ensure that the location constraint Equation 6.2c is met, we check if there are sufficient locations and maintain a counter representing the number of occupied locations at time t. Then, when selecting the set of valid times, if adding a job exceeds the max number of jobs in parallel constraint, the time is returned as invalid. Similarly, we want to reduce the action space further by pre-checking and filtering out by filtering invalid times and using a heuristic to select from the set of valid times.

Rather than selecting a time, job pair for an action during an MCTS rollout, an action consisting of a single job is selected using the rollout biasing. Using this job, the set of feasible times for the jobs is found (if any exist), and then a heuristic is used to select a single time from the list of times.

6.3.2.2 Heuristic for selecting allocation time

Using a heuristic to select the valid time for a candidate's job opens the question of what heuristic to use to make the selection. For this work, we consider two heuristics, the classic scheduling heuristic of 'earliest available time first' and a greedy heuristic which selects the time that has the highest expected resource budget, which we call 'greedy budget time'. The focus on these two heuristics is due to their differing strengths in selecting the time to allocate the job. The earliest time heuristic tries to allocate as many jobs as possible by finding the first possible time when it is feasible to schedule the job. However, this has no consideration for the stochastic and variable nature of the budget, which can result in leaving very little slack in the budget and can result in small variations, causing the schedule to be invalid. Alternatively, 'greedy budget time' tries to spread the risk of a schedule failing due to a poorly allocated job by scheduling it at a time when the budget is expected to be higher. By focusing on allocating to the safer times first, smaller variations in the production should cause schedules to fail less due to the less concentrated budget consumption.

It can be hard to select a heuristic that works for all possible distributions of job costs and resource production sequences, so in practice, the best feasible heuristic is a design decision for the practitioner. The most applicable heuristics should be selected by analysing their impact on representative data through backtesting.

6.3.3 Analysis

The stochastic resource scheduling problem is closely related to the classical scheduling problem on parallel machines with capacity constraints. This scheduling problem is known to be NP-Hard, and the stochastic resource problem has the additional complexity of resource uncertainty. The stochastic variant is also NP-Hard, and finding optimal solutions for large problems is infeasible. The search space for the stochastic resource problem is combinatorial between the joint problems of selecting a job, selecting a time, and selecting a testing location. This complexity limits the applicability of explicit solutions to only small-scale problems with a time horizon or very few jobs. Sampling-based methods are more applicable for larger practical real-world scaled problems but typically come with the trade-off of suboptimality. Monte Carlo Tree Search algorithms are sampling-based and are asymptotically optimal given enough randomness in their sampling and enough computation time.

The assumption that locations are homogenous and thus can be dropped from the action generation and replaced by a constraint check is restrictive and may not be feasible for all problems. For example, if robots need to travel to locations to perform a job, the location induces a state-dependent cost and cannot be considered homogenous. In this case, the state space reduction trick used in this method would not be valid, and the solution would need to consider the value of the location variable and include it in the cost computation, thus further expanding the state space.

Even with sampling, if the search space is massive, it can be difficult for the planner to converge to better solutions. The set of times that a job is valid for is small due to the combination of needing a valid location and, more difficultly, a point in time where the probability of a schedule failing due to production variability is below the risk threshold. Due to the arbitrary nature of the resource productions, there can not be a closed-form solution for when it is valid, which means that each point in time needs to be explicitly tested. This process allows the algorithm to know the expected utility of adding a specific job to an existing schedule, which is a far smaller search space than all action, job, and location combinations. This reduction of the action space to job-selection sequences with imputed times and locations results in faster convergence. However, there may be a reduction in optimality, but as we show in our simulation results for large problems, the effect on optimality is minimal while the scalability of the solution is heavily improved, allowing it to schedule far more jobs.

Robust scheduling requires producing schedules that have a near 0% chance of failure due to planning for the 'worst case'. Typically, the more restrictive the robustness constraints the more utility needs to be yielded to meet the constraint. If risk of failure can be quantified then it allows it to be tuned for operational purposes. Because production sequences have an associated probability an acceptable level of risk λ can be computed.

Bernoulli trials assume the events are independent which is not an accurate description of all time-varying stochastic processes. The difficulty is that there are two components of note, the amount of error and when the errors will occur. More complex temporal models are difficult especially when the underlying process is not possible to model explicitly. Hence, our method of estimating the expected amount of error and capturing a temporal structure using a tree allows a simple sampling and estimation of schedule risk.

6.4 Risk constrained probabilistic consumption scheduler for wind-powered hydrogen production

In this section, we validate our stochastic resource scheduling algorithm on a real-world instance of the problem. The problem we focus on in this work is scheduling scientific tasks using a limited hydrogen resource that is produced by stochastic wind power. We set up the background for the problem and then test our method in simulation to show that our algorithm is capable of producing high-quality results while limiting the operational risk for these operations.

DLR has a history of the application and production of hydrogen. The DLR Lampoldshausen site operates multiple test facilities and has expertise in space propulsion. There is currently a focus on transferring this expertise to terrestrial commercial applications and developing an environmentally friendly C02-neutral site. The purpose of this laboratory is the implementation, testing and demonstration of technologies related to the production, storage and use of hydrogen from renewable energy sources. Current hydrogen sources typically come from natural gas, whereas green hydrogen is produced from renewable (green) energy sources. With the focus on the reduction of C02 and environmental improvement, DLR wants to focus on the production and use of Green Hydrogen. Part of this process is the creation of the Zero Emission-H2 Test Center (H2CT), where green hydrogen is produced and consumed by engineering and scientific tests. The use of green hydrogen allows the test centre to facilitate the utilisation of hydrogen for scientific progress with a lower carbon footprint.

The test centre receives multiple requests, so there needs to be a process that allocates the limited resources of hydrogen and testing locations to these requests. These requests should be selected so the centre does not allocate more resources than it has. Additionally, the requests should be selected to maximise the reward, which is a measure of the expected utility to the test centre from the request, be it money, scientific impact, or some other factor. Due to dealing with external clients, the schedule of which requests will be serviced and when they need to be published in advance (around one week ahead) are hard to change. These constraints motivate producing robust (as in unlikely to fail) maximal schedules that are committed to for an upcoming time horizon. A schedule is considered to have failed if, during execution, there is not enough remaining hydrogen capacity to service the allocated requests.

The green hydrogen used by the centre is produced by electrolysis machines powered by renewable wind power sources. Wind power is a renewable source of electric power. However, its output is variable and uncertain on multiple time scales (hourly, daily, monthly, and yearly), which makes the scheduling complex. This uncertainty requires longer horizon robust schedule planning. To enable further green Hydrogen-based research, we want to establish a test centre that provides green hydrogen for engineering and scientific testing. The green hydrogen provided is produced through electrolysis powered by renewable power sources, specifically wind. The green hydrogen production process starts with the wind farm producing power from wind turbines, which is then used by electrolysers to produce gaseous hydrogen that is stored in tanks. Finally, the produced hydrogen is consumed by jobs at the test centre.

6.4.1 Simulation setup

This section outlines the setup for the simulation environment and the simulation results we use to empirically validate our algorithms' performance. The simulation comprises several components: the wind and electrolyser models, The data sets used for forecasting and ground truth, the jobs that need to be scheduled, the facility operation parameters, and finally, the methods used for comparison.

The wind farm produces power through the spinning of wind turbines. Turbine output is a function of the manufacturer's power curve and the wind speed at the hub height. The wind speed at different heights can be computed using $v_{wt} = v_m \frac{ln(h_m/z_0)}{ln(h_{wt}/z_0)}$. Where v_{wt} is wind velocity at the wind turbine, v_m is measured wind velocity, h_m is the height of the measurement, h_{wt} is the height of the wind turbine, and z_0 is the surface roughness which is estimated based on the type of terrain the wind farm is located in, (needs reference). A wind farm consists of multiple turbines so the production from one turbine P_{wt} is multiplied by the number of turbines n_{wt}. $P_f = P_{wt}.n_{wt}$ is the power produced from the wind farm given the wind speed at the hub height.

For the electrolyser the minimum required power to turn it on is E_{min} at which it produces its minimum output, and it follows a production curve until it reaches its maximum output at E_{max}. For this work, we assume E_{min}=E_{max} so the electrolyser turns on and produces its maximum effective output once the wind farm output reaches a sufficient level. The hydrogen produced for a given produced power is:

$$H(P_v) = \begin{cases} 0 & \text{if } P_f < E_{min} \\ 1 & \text{if } P_f \geq E_{min} \end{cases} \tag{6.6}$$

The hydrogen produced is then stored in the hydrogen storage tank and can be released out at a constant pressure to be used by the test centre. The tank has a maximum capacity hence resource budget $\mathcal{B}_{max}$, and a starting capacity $\mathcal{B}_0$.

For these simulations to model realistic wind conditions and the forecast data that was available at that point in the past, we used two meteorological research datasets. The forecast set is based on the TIGGEE Historical forecasts [264] With each week of data forecasted at two-hour blocks. For the ground truth set, we used the ERA5 historical data [265]. For both datasets, we are interested in a simulated facility located at Lampoldshausen, Germany (49.2631619N,9.4007921E).

For the simulations, we used a 7-day-ahead lock-in schedule requirement, which means that forecasts must be a week long. For simplicity, we did Monday to Monday, but the schedules do not need to be aligned with the week. For the hydrogen tank capacity, we set a maximum capacity of 30 units with a starting capacity of 20 units. In this work, we ignore the impact of temperature and changes in pressure. We treat the tank as a black box that can supply units of hydrogen at a known pressure.

To generate the set of jobs to be scheduled, each week 100 jobs with costs of 1 to 10 units were sampled from a uniform distribution $\mathbf{U}(1, 10)$ with an associated reward sampled from the uniform distribution 1 to 5 units $\mathbf{U}(1, 5)$. The length of jobs was drawn from a truncated normal distribution with a $\mathbf{N}(4, 2)$ with a lower bound cap of 1 hour and a maximum bound cap of 12 hours. This job distribution configuration was informed by discussions with DLR Lampoldshausen employees about the expected future demand of the planned systems. These simulations indicate the complexity of scheduling is expected to be faced in the future if the research infrastructure becomes more heavily used.

We compare our method against three baseline methods: forecast only (forecast), naively perturbing the forecasts (naive perturbation), and perfect forecasting (ground truth). The forecasts-only method produces a schedule using the forecast information and is unable to adjust the expectation of risk because it only has one scenario it considers. This is the same as solving a deterministic resource-constrained scheduling problem. Naive perturbation is the method of creating randomly perturbed forecasts by randomly perturbing the forecast and weighting all scenarios as equally likely. This method is useful to demonstrate if the process we use to perturb the forecasts and to allocate probability is beneficial compared to uninformed random perturbations. An illustrative example comparing our method to the naive perturbations is shown in Figure 6.2. Finally, perfect forecasting uses ground truth to create a schedule. While this method is extremely difficult to achieve in practice, it is useful to understand the relative performance of the methods by knowing how well they could have done if they had perfect information.

A 20-minute per trial computation limit for all of the methods compared. The naive perturbation method requires the selection of the number of forecast scenarios and the process used to perturb. We generated 200 perturbed forecasts to be scheduled with an equal weight assigned for each. We used the same Tau parameter for our PPV-based perturbation to perturb the schedules, flipped a biased coin for each timestep, and used that outcome to adjust the production.

To check the effect of heuristic selection on the methods, we produced schedules using an earliest-time heuristic and our greedy budget-time heuristic. The earliest-time heuristic simply finds the first point in time that a job can be validly scheduled and allocates it.

In these simulations, a failure is recorded for a trial if the schedule produced by the scheduler has a negative budget at any time when the produced schedule would have been executed using ground truth production.

Method	Heuristic	Confidence	N Fail	Success Rate	Reward	Failure
our method	earliest time	0.99	4	0.97(0.0)	85.12(8.8)	-0.5(4.2)
our method	greedy budget time	0.70	13	0.91(0.0)	90.33(8.4)	-1.8(8.4)
our method	greedy budget time	0.80	3	0.98(0.0)	87.83(7.6)	-0.3(2.2)
our method	greedy budget time	0.90	3	0.98(0.0)	87.80(7.8)	-0.2(1.9)
our method	greedy budget time	0.99	0	1.00(0.0)	81.82(7.3)	0.0(0.0)
forecast	earliest time	0.99	51	0.66(0.0)	93.37(10.3)	-5.2(10.2)
forecast	greedy budget time	0.99	30	0.80(0.0)	91.37(9.8)	-1.2(3.8)
ground truth	earliest time	0.99	0	1.00(0.0)	97.30(9.1)	0.0(0.0)
ground truth	greedy budget time	0.99	0	1.00(0.0)	94.25(9.3)	0.0(0.0)
naive perturb	earliest time	0.99	8	0.95(0.0)	85.71(11.6)	-1.6(7.8)
naive perturb	greedy budget time	0.70	9	0.94(0.0)	89.97(8.9)	-0.7(4.0)
naive perturb	greedy budget time	0.80	4	0.97(0.0)	89.45(8.8)	-0.1(0.7)
naive perturb	greedy budget time	0.90	3	0.98(0.0)	88.40(9.1)	-0.1(1.1)
naive perturb	greedy budget time	0.99	2	0.99(0.0)	83.66(11.8)	-0.0(0.5)

Table 6.1: Backtesting results for three years of weekly schedules. A schedule is invalid if the ground truth resource budget goes below 0. The mean for the reward and failure amounts is shown with their Standard deviation in brackets.

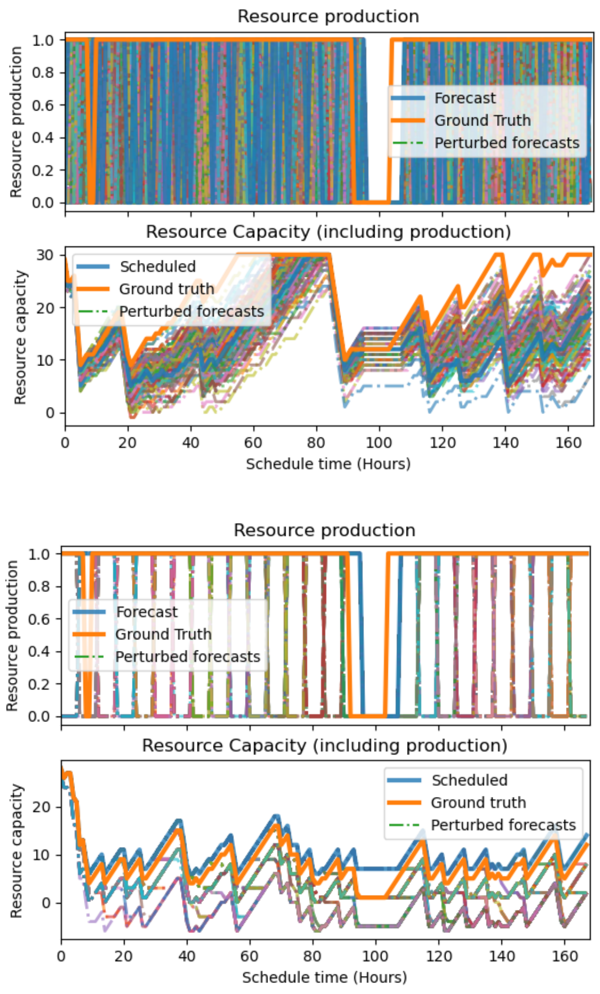


Figure 6.2: An example of a schedule produced using noisy perturbation forecasts (left) and our method to create forecasts(right). It can be seen that our method can create fewer forecasts with a more considered probability instead of the scattershot approach of noisily perturbing. This allows our method to create schedules that can use more of the resources with the same risk bounds.

6.4.2 Results

We ran our method and the comparative methods over each week of forecast data for January 2018 to January 2021, and the collated results are shown in Table 6.1. It can be seen that both forms of perturbation-based scenario schedulers were able to increase the success rate from 66%-80% to nearly 100%.

As the accepted level of risk increases, the perturbation methods' average reward increases with a corresponding increase in the number of failures. If the forecasts are accurate, then the acceptable level of risk can be set higher to gain a higher reward without as much downside.

Selection of scheduling heuristic

The *earliest time* heuristic has a higher failure rate because it front-loads schedules and makes them more sensitive to misestimation during the early part of the schedule. Additionally, by spending the expected amount of budget early, any negative error later in the schedule can cause a fail case without the possibility of any positive error covering it up.

The heuristic choice was less impactful for the perturbation-based methods, likely because they consider various scenarios. These scenarios likely include cases where the production error in later parts of the schedule would cause an allocation to fail and thus be invalid. The heuristic produces a different earliest time (if possible) for the different methods based on what is considered valid. This process results in schedules that could fail due to the greedy front loading being pruned due to the risk tolerance.

The *Greedy budget time* heuristic resulted in fewer errors at the expense of some potential reward. This was because spreading the jobs based on the highest minimum budget at each time uses the worst-performing forecast scenario at this time for picking when to allocate. This process is conservative and schedules fewer jobs because using the worst minimum results can result in having no more valid allocations once the worst schedule for each point in time has been exhausted. While there still may be valid allocations in individual scenarios, there is no valid allocation across the minimums for all times.

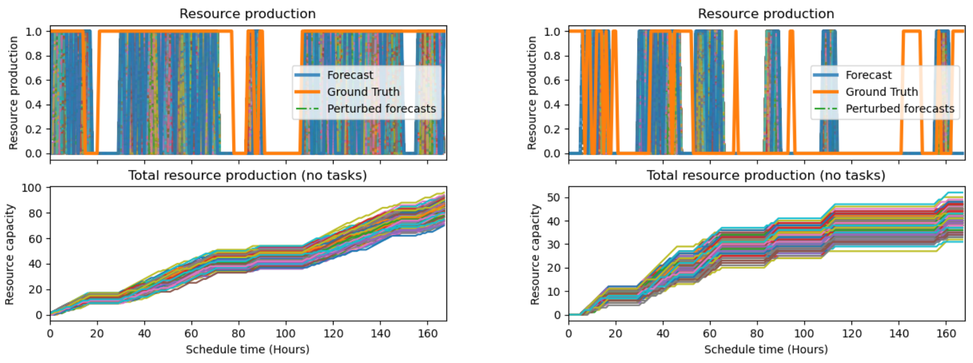


Figure 6.3: Fifty perturbed weekly forecasts with no consumption for January 2019 (left) and June 2019 (right). It can be seen that in this week in January, twice as much resource is produced than in July. This is due to the higher wind speeds running the electrolyser more often.

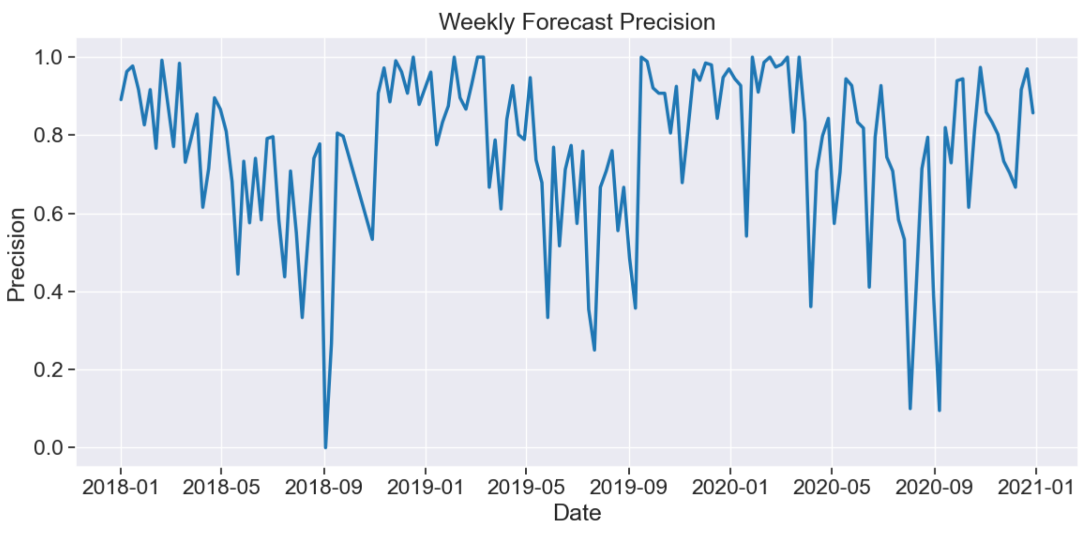


Figure 6.4: Weekly forecast precision over time. This plot shows that the precision of the wind speed forecast is highly seasonal.

Impact of precision parameter selection

The amount of refilling that happens during a schedule is a function of how often the wind strength is sufficiently high enough to run the electrolyser. The wind is highly seasonal; an illustrative example of two production sequences for different times of the year is shown in Figure 6.3. It can be seen that during January, it takes far longer than it does during July to fill the tank capacity. Knowing this fact, we investigated seasonality's effect on the forecasts' precision. Figure 6.4 shows how the precision of the forecast compared against the ground truth changes over time and has a seasonality. This is because the wind speed is lower during the less precise periods, so that the precision can be changed based on the season. When most of the wind speed is below the electrolyser turn-on point, then the number of true cases is lower. Additionally, any error in the magnitude causes the value to fluctuate around the turn-on point. From this, we infer that a combination of the lower number of true cases and the closeness to the turn-on point causes the lower precision.

This trend seems to hold over the long term for this wind forecasting problem. Knowing this fact allows the automatic selection of the precision parameter dependent on the time of year, allowing a variable error. This better-conditioned error allows tighter bounds and the appropriate allocation of risk, thus resulting in better results.

6.5 Summary

This chapter defines the risk-bounded stochastic resource scheduling problem, which involves scheduling jobs that consume a resource produced by an uncertain process. We propose a method to solve this problem by creating a set of perturbed forecasts from a single high-quality forecast of the resource production. These perturbed forecasts are created using a Bernoulli binary tree and the forecast's expected PPV. Using this set of forecasts, we use a Monte Carlo Tree Search algorithm to find schedules that maximise the utility and have a bounded risk. We apply this method to a real-world case study of scheduling operations for a green hydrogen research facility powered by wind power. We evaluate the performance of our method through simulation and compare it against other baseline methods. Overall, our method outperformed the baselines and reduced scheduling failures to near zero with limited loss of utility. This chapter presented the problem of scheduling with stochastic resources. The probabilistic consumption scheduling for wind-powered hydrogen production requires producing schedules based on consuming a stochastic resource. The uncertainty of resource levels is handled by our data-driven sampling-based model that produces schedules with a defined risk level. We create multiple alternative forecasts with associated probabilities to account for the uncertainty in resource production caused by imperfect forecasting. These forecasts are created using professional meteorological forecasts and a principled expectation of error based on external factors such as time of year. These forecasts can then be used to give the risk of a schedule failing, thus allowing us to plan a schedule with the level of acceptable risk. This process accounts for the uncertain level of the resources available and creates schedules that account for the probabilistic and temporal nature of the resource production. It is not just a matter of knowing how much resource is available; the timing and order of the operations using the resources must be considered.

Chapter 7

Conclusions and Future Work

The work in this thesis was done in the pursuit of addressing real-world practical multi-robot planning challenges in various domains. Real-world problems are human-driven, unstructured, uncertain and large-scale. We introduce a probabilistic scheduling framework for multi-robot coordination problems to deal with these complexities. Our framework considers four core sources of uncertainty (stochastic objective, constraints, tasks and resources) that each impact a typical component of a planning problem. Some of these components are not traditionally considered in coordination problems and algorithms. However, they have been considered in the field of scheduling. By viewing the multi-robot problems as stochastic scheduling problems, we can harness the strengths of both domains. This perspective illuminates the source of the complexities in the problem and can be used to identify potential simplifications. This understanding of the problem structure allows the development of more suitable algorithms than possible beforehand.

This thesis contributes to a suite of planning algorithms for multi-robot task allocation with our four core types of uncertainty in various scenarios. We emphasise scalable, practical solutions that would be feasible to deploy for real-world-sized problem instances. We explain the useful properties of our developed solutions analytically. Similarly, we demonstrate the effectiveness and quality of our solution against traditional approaches through simulation.

In this final chapter, we provide a short summary of each chapter in Section 7.1 with the main contributions of each in Section 7.2. Next, in Section 7.3 presents some possible future work directions. Finally, in Section 7.4, we conclude with an outlook for the future of multi-robot and automation applications in real-world scenarios.

7.1 Thesis summary

In this thesis, we presented multiple practical solutions to real-world multi-robot and automation planning problems. Our framework consists of viewing scheduling as uncertainty problems through decomposition, which gives insight into their structure. This insight is then useful in determining how best to approach the problem based on its scalability and type of uncertainty. We have demonstrated the usage of predictive techniques, decomposition or heuristics to account for the uncertainty and scale of the problem. We evaluated the methods we developed through algorithmic analysis, empirical simulation results, and hardware trials. These methods have been demonstrated to be useful in various domains, such as warehouse order picking, planetary exploration, oceanographic mapping and green hydrogen production scheduling.

7.1.1 Stochastic objectives: intuition-guided multi-robot orienteering (Chapter 3)

We first began in Chapter 3 by tackling problems with stochastic objectives. In this problem, the amount of utility gained by completing a task is not known a priori. We define the Discoverable Edge Cost Orienteering Problem (DECOP). In this problem, an agent traverses around an environment (with a noisy prior) to visit as many Points of Interest (POIS) as possible given a limited time or energy budget. Given the noisy map, the traversing cost may differ from the initial expected cost, thus changing the reward for a given action. To deal with this uncertainty, we replan as local information is gathered. However, naive replanning can be expensive computationally or lead to short-sighted reward-chasing behaviour.

We solve DECOP by modelling it as a different problem we call the Intuition-Guided Multi-robot Orienteering Problem (IGMOP). The IGMOP is similar to DECOP but has the additional reward of *intuition*, which means the robot gains the additional reward for following this intuition. We solve the IGMOP using a decentralized Monte Carlo Tree Search (Dec-MCTS) using a team consisting of a physical robot and virtual guiding agents (which provide the intuition). A ground station computes the virtual agents. Additionally, we speed up convergence between the two solutions by biasing the physical robot's search using pheromones created by the virtual agent search. This combination of biasing and DEC-MCTS allows the agents to communicate infrequently and solve the IGMOP problem while not over-constraining the physical agent's decision-making.

We provide empirical results for our formulation when applied in a planetary exploration context. We show that our method performs better than solving DECOP for the physical agent alone or using more extensive computing that does not take advantage of local information gained during plan execution.

A limitation of the Intuition-guided hybrid planner algorithm and IGMOP as a solution to the DECOP is their dependence on communication between the ground station and the mobile robot. Furthermore, we assume that the prior knowledge adequately represents the environment. If the prior is significantly inaccurate (we have not investigated the specific impact of levels of misalignment), the plans generated by the virtual agents may be detrimental to the mobile robots' performance rather than aiding in exploration. These limitations restrict the applicability of our solution for scenarios where communication and prior map information are limited, such as underground exploration. Future work could investigate the specific impacts of prior misalignment and address the issue by enabling the ground agent to model the benefit derived from the virtual agent's suggestions and dynamically adjusting the influence of their suggestions based on their alignment with the real world. Another area of potential research is to investigate stochastic representations of the map using bayesian methods, applying matching learning to estimate priors or other techniques used in recent advances in MCTS research to improve the performance of the MCTS component.

In conclusion, this chapter examines multi-robot planning and scheduling problems with stochastic objectives through a stochastic orienteering problem in the context of planetary exploration. We present a method to solve it that acknowledges the imbalance of computational resources and information available in a distributed heterogeneous team. We present simulation results that demonstrate that this is a promising approach towards enhancing the shared autonomy and efficiency of robotic exploration in uncertain environments.

7.1.2 Stochastic constraints: MVMF scheduler (Chapter 4)

In Chapter 4 we examine a type of scheduling problem with stochastic constraints. In this problem, the constraints determining whether a schedule is valid are stochastic and hard to define. We demonstrate this problem by creating a scheduler for a Multi-Vessel Multi-Float (MVMF) system operating in an oceanic environment. In this problem, the team must visit all points of interest while minimising makespan. The difficulty lies in the fact that the underactuated floats are subject to ocean currents and need to be collected within a time window. The ocean currents are chaotic and difficult to predict, which makes a schedule infeasible based on where the currents will send the floats. To address the chaos of the ocean currents, we assume the system is quasi-static and plans only short-horizon missions over a locally observed flow field.

To solve this problem, we developed a hierarchical solution that decomposes it into two smaller subproblems. The first problem that is solved is the minimal set of deployments that visit all of the POIs. Following this, the set of deployments is allocated to the vessels and floats through solving the second problem.

We show through simulation that our method outperforms a naive, greedy method. Our method can scale well with the size of the team and can use the additional agents efficiently as the team grows. Following this, we tested our method in the field using real hardware. Through these field trials, we validated our method and found practical areas to improve upon in future research. Finally, we extended upon a limitation that was observed in the first field trials through another brief set of field trials that included a replanning extension and iteratively updating the flowfield. This replanning extension is promising and needs further testing.

The core limitations of this work is caused by the difficulty of predicting dense flow fields of a dynamic ocean environment and the limited actuation of the bottom sensing floats. Prediction errors in the flow field can render the planned schedules infeasible. Furthermore, our framework requires the assumption of a quasi-static flow field, which, while valid for short durations in our field trials, may not always hold true. Future work can address this limitation by incorporating temporal flow field prediction models, online flow field re-estimation, and online schedule replanning. The fragility of the system can also be reduced by giving the bottom sensing floats some form of lateral actuation. However, this can be difficult to achieve and will increase the cost, so further research is required.

In summary, this chapter addressed problems with stochastic constraints in the context of a novel MVMF system operating in an ocean environment. By solving this problem, our framework can potentially reduce the barrier of entry for oceanographic data collection and motivate further research in dense flow field prediction and low-cost ocean robotics.

7.1.3 Stochastic tasks: dynamic warehouse order-picking (Chapter 5)

Scheduling problems with stochastic tasks were addressed in Chapter 5. This class of problem needs to find a schedule for tasks that are not known a priori and are created by some stochastic process. In this chapter, we focus on the dynamic warehouse order-picking problem. We show that vehicle routing approaches are limited in their scalability. To address this, we want to find a policy that maximises the throughput of the system.

We address the scalability by developing a computationally efficient partitioning-based algorithm that creates regions of equal work. We then present a method to compute this measure of work in each region and how to balance it between partitions iteratively. Using the partitions an agent is allocated to each and follows a simple service policy online to service the dynamically arriving demands.

We show analytically that servicing the regions of equal work is optimal for heavy-load regimes. Next, we empirically prove our method's performance by simulating a typical warehouse picking environment and comparing our method against a less specific dynamic vehicle routing method and a more naive method used in practice. Our results outperform these baseline methods and demonstrate the effectiveness of our proposed solutions.

While effective, the distributed equitable partition method has several limitations and assumptions. We assume an accurate prediction of the expected demands and no mismatch between the expected demand used to generate the partitions and the demand the actual demand the agents service. Future work can investigate the impact of such demand misalignment and techniques to dynamically adjust partitions online or reallocate agents or jobs between partitions to minimise this error. Furthermore, several sub-problems and considerations are overlooked, including multiple storage locations of items, multiple drop-off depots, finite shelf capacity, heterogenous item priority, and varying the size of items. These assumptions can impact the method's applicability to real-world warehousing operations and require further research to determine the impact that these constraints will have on our solution's usefulness.

In this chapter, we investigated scheduling with stochastic tasks, focusing on the dynamic warehouse order-picking problem where order arrivals and, thus, the tasks are uncertain. We demonstrated a technique that creates a policy that balances queues of expected tasks and indirectly optimises throughput. Our method offers a promising solution for improving the efficiency of warehouse order picking under the conditions of our problem formulation.

7.1.4 Stochastic resources: risk constrained scheduling with stochastic forecasts (Chapter 6)

The final class of scheduling problem in this thesis is scheduling with stochastic resources. In this problem, the resource consumed while servicing the tasks is replenished by a stochastic process. We show this problem in a real-world operational scenario of trying to schedule the usage of green hydrogen that is created via wind power. Wind is stochastic, so the amount of hydrogen is unknown.

To address this problem, we define the risk-constrained scheduling with stochastic forecasts problem. To solve this problem, we take a single high-quality forecast and an estimation of its precision, then use this to perturb it and create a set of sample forecasts. The perturbation is done by splitting the forecast into a series of windows and performing a Bernoulli trial for each. The outcome of this trial modifies the forecast during that window, and the final forecast has an associated probability based on the likelihood of those trial outcomes. Next, we take the set of forecasts and use a Monte Carlo tree search to find a schedule that maximises the utility of the scheduled jobs and has a bounded amount of risk.

We demonstrate our probabilistic Consumption Scheduler for Wind-Powered Hydrogen Production through simulation for a real-world client. We demonstrate that our method receives nearly as much utility as the baseline methods but with a drastically lower schedule risk.

Our method's key limitations lie in its reliance on a simplified error prediction model and simple infrastructure modelling. Future work could enhance the schedules' robustness and optimality by employing more sophisticated scenario-generation techniques or more informative perturbation methods beyond the simple Bernoulli method presented here. Additionally, investigating the impact of alternative problem parameterizations, such as its sensitivity to job generation functions or more complex hydrogen production and consumption models, could offer valuable research contributions.

In conclusion, this chapter introduces a complex stochastic scheduling problem and provides a method for producing high-quality schedules that are less likely to fail. Lastly, it also motivates further research in the crucial area of operationalizing green hydrogen infrastructure.

7.1.5 Handling general uncertainty

While this thesis's primary focus is on the design and applications of our probabilistic scheduling framework, it is crucial to understand how our methods deal with the two sources of general uncertainty, epistemic and aleatoric. Understanding these factors allows insight into our methods' limitations and directions for further improvement.

Epistemic uncertainty (incomplete knowledge) is primarily mitigated through prior models and beliefs that are updated with new information gathered during the planning or execution of a plan. Epistemic uncertainty is handled in the context of planetary exploration (Chapter 3); the robots begin with a noisy prior map (incomplete knowledge) and refine their understanding of the environment as they gather data during the mission. In Chapter 4, the MVMF framework briefly addresses this through the replanning process; this strategy addresses the epistemic uncertainty in the flow field temporal dynamics. Chapter 5 handles epistemic uncertainty in predicting future tasks by using historical data to construct a probabilistic demand.

Aleatoric uncertainty (inherent randomness) is managed by making the system robust to the variance and adapting the plan. Aletoric uncertainty is managed in Chapter 4 by accepting the randomness of flow fields and designing the constraints and operation window to minimise the exposure to the variance. The variability of dynamic orders in Chapter 5 is accounted for by the policy design and partitions, allowing the orders to be stochastic. Finally, in Chapter 6, the variability in wind power generation is addressed by generating multiple perturbed forecasts and incorporating risk constraints into the scheduling process, ensuring robustness against fluctuations in resource availability.

7.2 Summary of contributions

In this section, we elaborate on this thesis' specific contributions that are outlined in Chapter 1.4.

7.2.1 Multi-robot probabilistic scheduling framework

The novelty is that it is a new way of dealing with real-world and decision-making problems. The theoretical impact should lead to more research by people building upon the ideas presented. The broader impact is it should facilitate more field applications of robotics with a societal and economic impact. Solving this broad collection of examples leads to the beginning of a new theory of robotics scheduling that addresses a set of four fundamental components of how stochasticity manifests.

The four components of our probabilistic scheduling framework, stochastic objectives, constraints, tasks, and resources, must be handled differently. We develop new methods to handle these various conditions using our new insights gained through application.

Stochastic objective problems have fully known tasks, the constraints are static, and there are no uncertain resources to deal with. The main difficulty lies in producing a plan that has a high reward even though the utility of taking a sequence of actions is stochastic. We demonstrate solving a problem in this class by leveraging smart online replanning and integrating new information gained during operation to reduce the reward uncertainty of taking an action in the local area. Our method guided this local information gathering with a longer horizon search to avoid the myopia seen in local optimization methods.

For stochastic constraint problems, the tasks that need to be completed are fully known, the objective function is deterministic, and there are no stochastic consumable resources that need to be considered. However, the constraints that determine if a solution is valid can change, and the methods used to address this need to be able to deal with this. One type of method is the technique we employed to solve the MVMF problem. By understanding the physical process that changes the constraints and causes plans to become invalid, we produced a method that can successfully operate in this restrictive domain. The timescale of ocean current changes is large, meaning that if the plans are short enough, the ocean current appears to be quasi-static. This assumption allows our method to avoid the need for ocean current forecasting and makes planning under these conditions feasible.

In stochastic task problems, the objective function is deterministic, the constraints do not change during operation, and there are no stochastic resources. The uncertainty in the formulation of the task defines stochastic tasks. This task uncertainty can take multiple forms, such as not knowing ‘which’ tasks need to be done, ‘when’ the task will be completed, or ‘how’ to complete the task due to it having stochastic costs. In these problems, the focus is on creating methods that can account for tasks that will be required to complete in the future. This task uncertainty was dealt with by modelling a stochastic process that describes the future orders for an order-picking warehouse and then creating a policy (instead of a specific plan) that performs well given the operational reality described by that process. Maximising the system’s throughput described by that process means that our method accounts for many potential sequences of tasks rather than planning over what is available now. The key idea is to define the policy in a way such that all agents are working equally and that agents are near where work is expected to arrive, thus utilising the team effectively and reducing the cost of completing the work.

In a subset of scheduling problems, resources are required to complete tasks and are consumed during the work. The resources are considered stochastic when a stochastic process replenishes them, thus making it uncertain if enough resources will be available when completing a task in the future. The process of obtaining more resources can be uncertain for various reasons, such as unclear production amounts or arrival times. The uncertainty of resource levels is handled by our data-driven sampling-based model that produces schedules with a defined risk level. We create multiple alternative forecasts with associated probabilities to account for the uncertainty in resource production caused by imperfect forecasting. These forecasts are created using professional meteorological forecasts and a principled expectation of error based on external factors such as time of year. These forecasts can then be used to give the risk of a schedule failing, thus allowing us to plan a schedule with acceptable risk. This process accounts for the uncertain level of the resources available and creates schedules that account for the probabilistic and temporal nature of the resource production. It is not just a matter of knowing how much resource is available; the timing and order of the operations using the resources must be considered.

7.2.2 Intuition-guided hybrid planner algorithm

We developed a new paradigm for planning operations for low-power mobile agents. The novelty lies in the idea that the ground station can be used for more than monitoring and can be part of a heterogeneous team. This team consists of virtual guiding agents (computed by the ground station) and physical agents performing the work. This algorithm plans operations for the physical agent that are biased towards longer-term plans. The biasing for the physical agent's plans comes from integrating a virtual biasing function (created from the virtual agent's actions) that acts like an intuition for the ground agent. The biasing function is designed not to constrain the physical agent's decision-making ability, which allows it to utilise its local sensor information. It outperforms methods that only consider the mobile robot's limited computation or information available to the ground station. This is the first algorithm for solving orienteering by leveraging offboard computation in a way that alleviates myopia and does not constrain the mobile agent.

7.2.3 Hierarchical scheduling framework for Multi-Vessel-Multi-Float (MVMF) system

We proposed a hierarchical scheduling framework for a new multi-vessel multi-float system. The MVMF problem involves multiple surface vessels picking up and deploying multiple information-gathering floats in an oceanic environment. The novelty of the MVMF system is that it is a new approach towards oceanographic information gathering with a substantially lower hardware cost, but it comes with new planning complexities. To address this, we formulated a new problem definition that defines the operation of these systems and introduced new algorithms capable of producing operation schedules. Our method is a sampling-based two-step hierarchical algorithm that first finds a subset of deployment locations. It then efficiently searches for pickup and drop-off actions that visit the deployment locations. Our algorithm offers a scalable way to find a plan that utilises all the agents and meets the demands of scheduling operations for a MVMF system with the stochastic constraints caused by changing ocean currents. Additionally, it is fast enough to compute that it can be used to perform online replanning.

7.2.4 Distributed partitioning based distributed multi-agent warehouse order picking

We propose an equitable partitioning and batch servicing policy-based approach to solve the dynamic warehouse order-picking problem. This method includes a novel cost metric that accounts for the obstacles and depot returns in warehouse order picking. It also includes approximations to compute it in polynomial time for an arbitrary distribution that describes orders. This policy has been shown analytically to perform well in heavy-loading scenarios. Similarly, we demonstrate empirically that our method has improved performance compared to typical order picking and a more generic equitable partitioning method.

7.2.5 Forecast perturbation based risk constrained scheduling

We developed a new perturbation-based risk-constrained scheduling method that can find risk-bounded solutions for scheduling problems with uncertain resource levels. This is the first algorithm that considers the problem of the resource consumed by the schedule being replenished by a stochastic process. The algorithm is a data-driven sampling-based method that leverages existing high-quality forecasts as a prior. It avoids the problem of trying to model complex systems and uses existing expertise. A single parameter for error parameterises the uncertainty and produces schedules with a bounded risk level. This approach allows the scheduling of medium-term operations that consume a stochastic resource produced by an arbitrarily complex stochastic process.

7.2.6 Analytical results

For each method we presented in this thesis, we have also provided a set of analytical results. These results show our developed methods' suitability, scalability and usefulness for their designated problems. Despite the computational complexity of the addressed problems, we show that our methods can produce high-quality solutions that are practical and suitable for real-world applications.

Our analysis for the intuition-guided hybrid planner algorithm focused on showing that a good solution to the IGMOP was also a good solution to the DECOP. We show that the deviations between the agent's solutions for the two problems are caused by the mismatch in information between the two agents. Given that the ground station has substantially more computational power, the alternative plans produced by the mobile robot must have enough marginal benefit that sticking to the intuition would be suboptimal. Similarly, we show that the biasing function we introduce to the Dec-MCTS component does not break the asymptotically optimal property, as every action can still be explored during the planning phase.

The main analytical results for MVMF scheduler are to show that the hierarchical method produces valid solutions to the MVMF problem in a scalable way. We first show that the large search space is computationally difficult, and naive approaches are intractable. Next, we show that it is not possible for the planner to consider invalid actions; thus, if a plan is produced, it is valid with respect to the constraints. Finally, we show that even though we cannot prove the optimality of our algorithm. Through using the physical properties of ocean currents, we prove that the joint solution produced by our method is near optimal.

There are several analytical contributions to our distributed equitable partitioning policy algorithm. In particular, we show using queueing theory for heavy-loading regimes, and throughput can be maximised by creating partitions that balance the cost to service. Next, we show that our expected workload heuristic can be computed in polynomial time despite containing a TSP as part of the cost. Finally, we show that it is scalable by avoiding considering the pairwise combination of all partitions, simplifying the task allocation.

Our analysis for the risk-constrained scheduling with stochastic forecasts algorithm shows the problem is NP-hard, which inherently limits the applicability of exact methods and motivates sampling-based solutions. Our analysis also provides insight into the balance between the level of risk that needs to be accepted and the utility of produced solutions.

7.2.7 Empirical results

For each method, we presented empirical results from extensive simulation trials. These simulation results validate the theoretical claims and demonstrate the applicability of our algorithms to their designed use cases. For all algorithms, the computation is shown to be fast and scalable to real-world-sized instances. These simulation experiments were performed for scenarios that match real-world problems.

For our intuition-guided hybrid planner algorithm, we presented experiments demonstrating its ability in a planetary exploration context. These results showed that our algorithm improved the number of visited points over the ‘solo’ planner (consisting of the mobile agent only) and the ‘long horizon’ planner that used the ground stations compute only. These improvements were achieved using limited communication and did not require additional computational power on the mobile robot.

For the MVMF scheduling algorithm, we present an extensive set of simulations in various problem instances. These results empirically confirm our analytical claims and show that our method outperforms a greedy sequential allocation method. These simulation results also show the scalability of our approach both in terms of effectively utilizing more agents and being fast to compute as team size grows.

We empirically evaluated our distributed equitable partitioning policy algorithm for dynamic warehouse order picking through simulation. These simulations present results for two months of operation in a typically sized warehouse environment. Our results demonstrate that our approach can service more orders than the typical naive picklist approach and a less considered partitioning-based method. We show that the importance of partitioning grows as a function of picker capacity due to the larger benefit of being able to minimise travel between items increases. We also present evidence that our equal work partitioning algorithm creates partitions that are approximately equal concerning the partition cost metric.

Through backtesting using years of historical weather data, we empirically show the quality of our risk-constrained scheduling with a stochastic forecast algorithm. Our results show that our method achieves results with nearly as much utility as the forecast-only scheduler but with near-zero failure cases. Additionally, the simulations show that our method respects the confidence bounds and looser confidence results in higher utility with more failures and conversely. Similarly, we show that the precision of the forecast is dependent on the time of year.

7.2.8 Hardware field trials of the MVMF system

Hardware field experiments were conducted to assess the practicality and performance of our MVMF scheduling algorithm. The trials confirmed that the system operates under the anticipated constraints in a real-world oceanic environment. Moreover, these real-world experiments provided critical insights that inform future research and enhancements to our algorithm and other marine robotics research.

Some of the main insights gained were regarding how to run hardware operations in the water that were not initially considered. A key oversight was not accounting for the wake of the deployment vessel when deploying the floats. Other insights are about the limitations of 2D ocean current modelling assumptions being broken by observing 3D current structures such as Langmuir circulation. Finally, while the ocean currents are slow to change, integrating new information through replanning can account for accumulating estimation errors over time. This idea was tested in a short exploratory field trial that integrated a simple replanning extension. This replanning was shown to be promising and motivates further research.

7.3 Future work

Throughout this work and the analysis of the related literature, there are several possible general directions for future work.

7.3.1 Improvements to flow field estimation

The flow field estimation used in our MVMF field trials in Section 4.5.1 works under the assumption that flows of water are incompressible, operate in a 2D plane and have no interaction with physical obstacles. These assumptions are reasonable in deep water operations, but in our field trials, we have observed these not holding true for shallower waters for several reasons. To improve the robustness of the estimator, it is important to consider the following factors: identify where the 2D flow incompressibility assumption fails and deal with these cases, account for the interaction between the bathymetry and the flow fields, and account for wind conditions at the surface. Extending the method to deal with these issues should improve the flow estimation quality and enable our scheduler to produce better plans.

Another area of operational improvement is considering the initial flow field estimation calibration dives as part of the sampling mission. Our current operation plan is to deploy the floats spatially dispersed around the environment for short dives so they will resurface within eyesight. The surfacing location of the floats is then used to calibrate the flow fields, and the planner can now produce the information-gathering mission plans. Rather than sampling randomly, we could assume a random or zero initial prior flow field and select initial positions most likely to gather useful information, thus reducing the amount of work remaining after the calibration phase. The idea would be to gradually build a full-flow field map iteratively, with the estimation getting more and more accurate until it is sufficient for planning.

7.3.2 Online adaptive repartitioning

The equitable partitioning method in Section 5.3.2 is done offline, and the allocations for the items to be picked are done online. For future work, there is potential to update the partitions online using iterative optimisation if the work balance becomes too unequal. A mismatch between the historical data and the realised order distribution can cause an imbalance in the work required to service the partitions. The orders can be reallocated by updating the partitions online, and the system can be rebalanced to match the changed distribution. This is not a simple extension and requires further research, as the changing regions require more trajectory planning to avoid collisions between agents. Additionally, careful consideration needs to be made not to cause orders to switch back and forth between agents, which can result in items being never serviced.

7.3.3 Learning approximation functions

There has been a large amount of research and advances in the abilities of deep learning. While deep learning and reinforcement learning have yet to be demonstrated to be useful for many planning-type problems, there are still directions where they can be used beneficially as part of a planning algorithm. The main avenue we believe is promising is learning approximation functions that can be used as heuristics or predictors for specific problems. It could be possible to speed up some problems through deep learning, reinforcement learning or imitation learning. Using the models as a function inside the planning framework can lead to higher-quality results or speed up convergence to high-quality solutions. Specific examples would be to learn to predict the map based on the local observations for the planetary exploration problem or learn a function that can estimate the risk of job time allocations to augment the scheduling in the risk-constrained resource allocation problem.

7.3.4 Effects of compound types of stochastic uncertainty

This work is an initial investigation into the probabilistic scheduling framework. In this work, each of the components was considered separately to be able to identify key features and gain a deeper understanding of their implications. For future work, we would like to examine the impact of compounding multiple types of uncertainty together for more complex and uncertain problems.

We provide a few examples related to the domains addressed in this work for discussion, but our methods can apply to many domains. An example of a stochastic resource and constraint problem could be scheduling a MVMF system where the vessels are charged using solar panels on the surface vessels. A stochastic resource and task example is scheduling wind-powered green hydrogen consumption, where the demands arrive during the week and must be added to the schedule. For stochastic tasks and constraints, you could consider a problem where warehouse operations have dynamic tasks, but the machines can break down during operation, and the goal is to find a schedule that accounts for these possible failures. Finally, for stochastic objectives and tasks, the problem could be exploring an unknown area with tasks that are revealed during exploration or dynamically allocated to the agents. The possibilities are not limited to these examples, as the stochastic components can come in many variations and mixtures.

By viewing the multi-robot problems as stochastic scheduling problems, we can harness the strengths of both domains. This perspective illuminates the source of the complexities in the problem and can be used to identify potential simplifications. By furthering the understanding of these problem classes and their interactions, we can feasibly address even more real-world problems and extend the abilities of multi-agent systems.

Furthermore, there exists a body of work in uncertainty quantification and identifying the structure of uncertainty. Considering additional characteristics of uncertainty in more detail, such as temporal dynamics and distribution shifts, could lead to improvements in the uncertainty identification of our framework and give insight into techniques for effectively managing uncertainty in multi-robot planning problems.

7.3.5 Hardware experiments

In this thesis, we did hardware experiments for one of the four tested problem domains. We demonstrated that including hardware trials as part of the research process allows for identifying potential weaknesses and leads to more suitable algorithms for real-world applications. We presented empirical results from simulated experiments for the rest of our methods. These simulations are useful for validating the behaviour and suitability of our algorithms to solve the problem as formulated. However, simulations do not always capture the complexities and realities of operating in the real world. These problems can be sources of uncertainty not accounted for in the algorithm, communication failures, and other environmental limitations. As such, we would like to demonstrate our other methods onboard real systems and extend our work into field applications.

7.4 Outlook

In Chapter 1, we argued that robotics are important, and there have been many practical advances in hardware and algorithms. A single robot can only do so much, and to deal with large, complicated problems that matter to people, you need to use multi-robots. Similarly, the adoption of Industry 5.0 presents many opportunities for robotics and will be transformative for society. This new paradigm is characterised by flexibility, robustness, and collaboration between humans and machines. Integrating smarter probabilistic scheduling into multi-agent systems will enable greater flexibility, reliability, sustainability, and efficiency in systems that can handle real-world problems.

In the near future, we can imagine the widespread emergence of 'smart factories' where humans and robots work in tandem. These new factories can leverage the abilities of robots and humans while adapting to the rapidly changing demands of modern society. Probabilistic scheduling that accounts for the messiness and complexities of the real world will enable these multi-agent systems to adjust to changing demands and the dynamic nature of human-centric environments. Moreover, probabilistic scheduling will play a crucial role in optimising energy consumption and minimising waste by more efficiently using resources.

The usefulness of Multi-agents is not restricted to business operations. Making robotics more accessible and functional can improve many aspects of scientific research. Through the advances in system robustness and scalability afforded by probabilistic scheduling, new avenues of field research can be performed. Some new problems that could be feasible are large-scale information gathering, exploring unknown planetary bodies autonomously, and continuously monitoring forests for potential fire risks. By enabling more general scientific research, improvements in robotics can improve society in general.

The research presented in this thesis lays the groundwork for these advancements. Our work offers a glimpse into a future where robots are general enough to deal with the uncertainty and multiple paradigms of structure that form a sort of ordered chaos that we humans operate in every day. This future is about enhancing human life and steering society towards a more sustainable, efficient and connected future.

Bibliography

[1] Craig Webster and Stanislav Ivanov. *Robotics, artificial intelligence, and the evolving nature of work*. Springer, 2020.

[2] M. Dunbabin and L. Marques. Robots for Environmental Monitoring: Significant Advancements and Applications. *IEEE Robotics Automation Magazine*, 19(1):24–39, March 2012. ISSN 1070-9932. doi: 10.1109/MRA.2011.2181683.

[3] Daniele Calisi, Fabio Cottefoglie, Lorenzo D'Agostini, Francesca Giannone, Fabrizio Nenci, Paolo Salonia, Marco Zaratti, and Vittorio Amos Ziparo. Robotics and virtual reality for cultural heritage digitization and fruition. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42, 2017.

[4] Michela Cigola, Arturo Gallozzi, Luca James Senatore, and Roberto Di Maccio. The heritagebot project. putting robotics into survey. In *New Activities For Cultural Heritage: Proceedings of the International Conference Heritagebot*, pages 113–121. Springer, 2017.

[5] David Ball, Patrick Ross, Andrew English, Tim Patten, Ben Upcroft, Robert Fitch, Salah Sukkarieh, Gordon Wyeth, and Peter Corke. Robotics for sustainable broad-acre agriculture. In *Proceedings of Field and service robotics*, pages 439–453. Springer, 2015.

[6] J. P. Underwood, M. Calleija, Z. Taylor, C. Hung, J. Nieto, R. Fitch, and S. Sukkarieh. Real-time target detection and steerable spray for vegetable crops. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2015.

[7] Melanie Birks, Marie Bodak, Joanna Barlas, June Harwood, and Mary Pether. Robotic seals as therapeutic tools in an aged care facility: a qualitative study. *Journal of aging research*, 2016.

[8] Ivar Mendez, Michael Jong, Debra Keays-White, and Gail Turner. The use of remote presence for health care delivery in a northern inuit community: a feasibility study. *International journal of circumpolar health*, 72(1):21112, 2013.

[9] Angie Abdilla and Robert Fitch. Fcj-209 indigenous knowledge systems and pattern thinking: An expanded analysis of the first indigenous robotics prototype workshop. *The Fibreculture Journal*, 2017.

[10] Oluwadara Asinobi, Janet Allison, Martin McKinney, Siobhan Flynn, Michaela Black, and Adrian Moore. Empirical findings: The use of robotics to engage the youth from lower socio-economic areas. In *Proceedings of IEEE International Symposium on Technology and Society (ISTAS)*, pages 1–6. IEEE, 2015.

[11] Nikola Gjeldum, Amanda Aljinovic, M Crnjac Zizic, and Marko Mladineo. Collaborative robot task allocation on an assembly line using the decision support system. *International Journal of Computer Integrated Manufacturing*, 35(4-5):510–526, 2022.

[12] Aaron D Neal, Richard G Sharpe, Katherine van Lopik, James Tribe, Paul Goodall, Heinz Lugo, Diana Segura-Velandia, Paul Conway, Lisa M Jackson, Thomas W Jackson, et al. The potential of industry 4.0 cyber physical system to improve quality assurance: An automotive case study for wash monitoring of returnable transit items. *CIRP Journal of Manufacturing Science and Technology*, 32:461–475, 2021.

[13] Xun Xu, Yuqian Lu, Birgit Vogel-Heuser, and Lihui Wang. Industry 4.0 and industry 5.0—inception, conception and perception. *Journal of Manufacturing Systems*, 61:530–535, 2021.

[14] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008.

[15] Nils Boysen, Stefan Schwerdfeger, and Felix Weidinger. Scheduling last-mile deliveries with truck-based autonomous robots. *European Journal of Operational Research*, 271(3): 1085–1099, 2018.

[16] Alena Otto, Niels Agatz, James Campbell, Bruce Golden, and Erwin Pesch. Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks*, 72(4):411–458, 2018.

[17] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert systems with applications*, 165:113816, 2021.

[18] Gourav Bathla, Kishor Bhadane, Rahul Kumar Singh, Rajneesh Kumar, Rajanikanth Aluvalu, Rajalakshmi Krishnamurthi, Adarsh Kumar, RN Thakur, and Shakila Basheer. Autonomous vehicles and intelligent automation: Applications, challenges, and opportunities. *Mobile Information Systems*, 2022(1):7632892, 2022.

[19] Daniel J Fagnant and Kara Kockelman. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. *Transportation Research Part A: Policy and Practice*, 77:167–181, 2015.

[20] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2012.

[21] Alessandro Farinelli, Luca Iocchi, and Daniele Nardi. Multirobot systems: a classification focused on coordination. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 34(5):2015–2028, 2004.

[22] Lea Hallik, Egidijus Šarauskis, Marius Kazlauskas, Indrė Bručienė, Gintautas Mozgeris, Dainius Steponavičius, and Toomas Tõrra. Proximal sensing sensors for monitoring crop growth. In *Information and Communication Technologies for Agriculture—Theme I: Sensors*, pages 43–97. Springer, 2022.

[23] Michael Warren, Peter Corke, Oscar Pizarro, Stefan Williams, and Ben Upcroft. Visual sea-floor mapping from low overlap imagery using bi-objective bundle adjustment and constrained motion. In *Proceedings of Australasian Conference of Robotics and Automation (ACRA)*, pages 1–10, 2012.

[24] Australasian transport news. Patrick straddle carriers image, 2021. URL `https://assets.primecreative.com.au/imagegen/cr/340/255/s3/cougar-assets/atnau/2021/02/08/Misc/Patrick1.jpg`. [Online; accessed November 07, 2021].

[25] Raffaello D'Andrea. Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead. *IEEE Transactions on Automation Science and Engineering*, 9(4):638–639, 2012.

[26] Giovanni D'urso, James Ju Heon Lee, Oscar Pizarro, Chanyeol Yoo, and Robert Fitch. Hierarchical mcts for scalable multi-vessel multi-float systems. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 8664–8670. IEEE, 2021.

[27] Giovanni D'Urso, Stephen L Smith, Ramgopal Mettu, Timo Oksanen, and Robert Fitch. Multi-vehicle refill scheduling with queueing. *Computers and electronics in agriculture*, 144:44–57, 2018.

[28] Hugh Durrant-Whyte, Daniel Pagac, BEN Rogers, Michael Stevens, and Graeme Nelmes. Field and service applications-an autonomous straddle carrier for movement of shipping containers-from research to operational autonomous systems. *IEEE Robotics & Automation Magazine*, 14(3):14–23, 2007.

[29] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[30] Ko-Wan Tsou, Yu-Ting Hung, and Yao-Lin Chang. An accessibility-based integrated measure of relative spatial equity in urban public facilities. *Cities*, 22(6):424–435, 2005.

[31] Eugene L Lawler. The traveling salesman problem: a guided tour of combinatorial optimization. *Wiley-Interscience Series in Discrete Mathematics*, 1985.

[32] Silvano Martello and Paolo Toth. *Knapsack problems: algorithms and computer implementations*. John Wiley & Sons, Inc., 1990.

[33] Timothy Patten, Robert Fitch, and Salah Sukkarieh. Large-scale near-optimal decentralised information gathering with multiple mobile robots. In *Proceedings of Australasian Conference of Robotics and Automation (ACRA)*, pages 0–15, 2013.

[34] Paolo Toth and Daniele Vigo. *The vehicle routing problem*. SIAM, 2002.

[35] Daniel Alejandro Rossit, Fernando Tohmé, and Mariano Frutos. Industry 4.0: smart scheduling. *International Journal of Production Research*, 57(12):3802–3813, 2019.

[36] Wei Xiang and Heow Pueh Lee. Ant colony intelligence in multi-agent dynamic manufacturing scheduling. *Engineering Applications of Artificial Intelligence*, 21(1):73–85, 2008.

[37] Tian Zhang, Wei Chen, Zhu Han, and Zhigang Cao. Charging scheduling of electric vehicles with local renewable energy under uncertain electric vehicle arrival and grid power price. *IEEE Transactions on Vehicular Technology*, 63(6):2600–2612, 2013.

[38] Randolph W Hall et al. *Handbook of healthcare system scheduling*. Springer, 2012.

[39] Sönke Hartmann. A competitive genetic algorithm for resource-constrained project scheduling. *Naval Research Logistics (NRL)*, 45(7):733–750, 1998.

[40] Javier Alcaraz and Concepción Maroto. A robust genetic algorithm for resource allocation in project scheduling. *Annals of operations Research*, 102:83–109, 2001.

[41] Willy Herroelen and Roel Leus. Project scheduling under uncertainty: Survey and research potentials. *European journal of operational research*, 165(2):289–306, 2005.

[42] Michael Pinedo and Khosrow Hadavi. Scheduling: theory, algorithms and systems development. In *Operations Research Proceedings 1991: Papers of the 20th Annual Meeting/Vorträge der 20. Jahrestagung*, pages 35–42. Springer, 1992.

[43] Ronald Lewis Graham, Eugene Leighton Lawler, Jan Karel Lenstra, and AHG Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In *Annals of discrete mathematics*, volume 5, pages 287–326. Elsevier, 1979.

[44] Andrea Casalino, Eleonora Mazzocca, Maria Grazia Di Giorgio, Andrea Maria Zanchettin, and Paolo Rocco. Task scheduling for human-robot collaboration with uncertain duration of tasks: a fuzzy approach. In *2019 7th International Conference on Control, Mechatronics and Automation (ICCMA)*, pages 90–97. IEEE, 2019.

[45] Filipe Rodrigues and Agostinho Agra. An exact robust approach for the integrated berth allocation and quay crane scheduling problem under uncertain arrival times. *European Journal of Operational Research*, 295(2):499–516, 2021.

[46] Francesco Bullo, Emilio Frazzoli, Marco Pavone, Ketan Savla, and Stephen L Smith. Dynamic vehicle routing for robotic systems. *Proceedings of the IEEE*, 99(9):1482–1504, 2011.

[47] Bo Fu, William Smith, Denise M Rizzo, Matthew Castanier, Maani Ghaffari, and Kira Barton. Robust task scheduling for heterogeneous robot teams under capability uncertainty. *IEEE Transactions on Robotics*, 39(2):1087–1105, 2022.

[48] Olivier Lambrechts, Erik Demeulemeester, and Willy Herroelen. Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. *Journal of scheduling*, 11(2):121–136, 2008.

[49] Mohammad Fathi and Hassan Bevrani. Adaptive energy consumption scheduling for connected microgrids under demand uncertainty. *IEEE Transactions on Power Delivery*, 28(3): 1576–1583, 2013.

[50] Xiaoqing Xu, Wentian Cui, Jun Lin, and Yanjun Qian. Robust makespan minimisation in identical parallel machine scheduling problem with interval data. *International Journal of Production Research*, 51(12):3532–3548, 2013.

[51] V Jorge Leon, S David Wu, and Robert H Storer. Robustness measures and robust scheduling for job shops. *IIE transactions*, 26(5):32–43, 1994.

[52] Sarah Vanheusden, Teun Van Gils, An Caris, Katrien Ramaekers, and Kris Braekers. Operational workload balancing in manual order picking. *Computers & Industrial Engineering*, 141:106269, 2020.

[53] Pranab K Muhuri and Kaushal K Shukla. Real-time task scheduling with fuzzy uncertainty in processing times and deadlines. *Applied soft computing*, 8(1):1–13, 2008.

[54] Ali Forootani, Raffaele Iervolino, Massimo Tipaldi, and Joshua Neilson. Approximate dynamic programming for stochastic resource allocation problems. *IEEE/CAA Journal of Automatica Sinica*, 7(4):975–990, 2020.

[55] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International journal of robotics research*, 23(9):939–954, 2004.

[56] Veniamin Tereshchuk, John Stewart, Nikolay Bykov, Samuel Pedigo, Santosh Devasia, and Ashis G Banerjee. An efficient scheduling algorithm for multi-robot task allocation in assembling aircraft structures. *IEEE Robotics & Automation Letters*, 4(4):3844–3851, 2019.

[57] Matthew C. Gombolay, Ronald J. Wilcox, and Julie A. Shah. Fast scheduling of robot teams performing tasks with temporospatial constraints. *IEEE Transactions on Robotics*, 34(1): 220–239, 2018. doi: 10.1109/TRO.2018.2795034.

[58] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[59] Micah Corah and Nathan Michael. Efficient online multi-robot exploration via distributed sequential greedy assignment. In *Robotics: Science and Systems (RSS)*, volume 13. Cambridge, MA, 2017.

[60] Brian P Gerkey and Maja J Mataric. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[61] Armin Sadeghi and Stephen L. Smith. Heterogeneous task allocation and sequencing via decentralized large neighborhood search. *Unmanned Systems*, 5(2):79–95, 2017.

[62] Geoffrey A Hollinger and Gaurav S Sukhatme. Sampling-based robotic information gathering algorithms. *The International Journal of Robotics Research*, 33(9):1271–1287, 2014.

[63] Guillaume Sartoretti, Yue Wu, William Paivine, TK Satish Kumar, Sven Koenig, and Howie Choset. Distributed reinforcement learning for multi-robot decentralized collective construction. In *Proceedings of Distributed Autonomous Robotic Systems: The International Symposium*, pages 35–49. Springer, 2019.

[64] Benjamin Riviere, Wolfgang Hönig, Matthew Anderson, and Soon-Jo Chung. Neural tree expansion for multi-robot planning in non-cooperative environments. *IEEE Robotics and Automation Letters*, 6(4):6868–6875, 2021.

[65] Xiangyu Yue, Yang Zhang, Sicheng Zhao, Alberto Sangiovanni-Vincentelli, Kurt Keutzer, and Boqing Gong. Domain randomization and pyramid consistency: Simulation-to-real generalization without accessing target domain data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 2100–2110, 2019.

[66] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S Davis, and Tom Goldstein. Universal adversarial training. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5636–5643, 2020.

[67] Ting-Wu Chin, Cha Zhang, and Diana Marculescu. Renofeation: A simple transfer learning method for improved adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3243–3252, 2021.

[68] Chong Zhou and Randy C Paffenroth. Anomaly detection with robust deep autoencoders. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 665–674, 2017.

[69] Adrien Baranes and Pierre-Yves Oudeyer. R-iac: Robust intrinsically motivated exploration and active learning. *IEEE Transactions on Autonomous Mental Development*, 1(3):155–169, 2009.

[70] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021.

[71] Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022.

[72] Jiachen Li, Hengbo Ma, and Masayoshi Tomizuka. Interaction-aware multi-agent tracking and probabilistic behavior prediction via adversarial learning. In *2019 international conference on robotics and automation (ICRA)*, pages 6658–6664. IEEE, 2019.

[73] Yuming Zhang, Bohao Feng, Wei Quan, Aleteng Tian, Keshav Sood, Youfang Lin, and Hongke Zhang. Cooperative edge caching: A multi-agent deep learning based approach. *IEEE Access*, 8:133212–133224, 2020.

[74] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[75] Lars Blackmore and Masahiro Ono. Convex chance constrained predictive control without sampling. In *Proceedings of AIAA guidance, navigation, and control conference*, page 5876, 2009.

[76] Daniele Bernardini and Alberto Bemporad. Scenario-based model predictive control of stochastic constrained linear systems. In *Proceedings of IEEE Conference on Decision and Control (CDC)*, pages 6333–6338. IEEE, 2009.

[77] Lars Blackmore, Masahiro Ono, Askar Bektassov, and Brian C Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE Transactions on Robotics*, 26(3):502–517, 2010.

[78] Haifeng Qiu, Wei Gu, Pengxiang Liu, Qirun Sun, Zhi Wu, and Xi Lu. Application of two-stage robust optimization theory in power system scheduling under uncertainties: A review and perspective. *Energy*, page 123942, 2022.

[79] Miguel A Salido, Joan Escamilla, Adriana Giret, and Federico Barber. A genetic algorithm for energy-efficiency in job-shop scheduling. *The International Journal of Advanced Manufacturing Technology*, 85:1303–1314, 2016.

[80] Maani Ghaffari Jadidi, Jaime Valls Miro, and Gamini Dissanayake. Sampling-based incremental information gathering with applications to robotic exploration and environmental monitoring. *The International Journal of Robotics Research*, 38(6):658–685, 2019.

[81] AJRM Gademann, Jeroen P Van Den Berg, and Hassan H Van Der Hoff. An order batching algorithm for wave picking in a parallel-aisle warehouse. *IIE Trans.*, 33(5):385–398, 2001.

[82] Giovanni D'urso, Armin Sadeghi, Chanyeol Yoo, Stephen L Smith, and Robert Fitch. Distributed multi-robot equitable partitioning algorithm for allocation in warehouse picking scenarios. In *Proceedings of IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1–8. IEEE, 2023.

[83] Giovanni D'urso, James Ju Heon Lee, Ki Myung Brian Lee, Jackson Shields, Brenton Leighton, Oscar Pizarro, Chanyeol Yoo, and Robert Fitch. Field trial on ocean estimation for multi-vessel multi-float-based active perception. *arXiv preprint arXiv:2106.09279*, 2021.

[84] Giovanni D'urso, Michael Füting, and Robert Fitch. Probabilistic consumption scheduler for wind-powered hydrogen production. *Available at SSRN 4766754*.

[85] Ioannis Rekleitis, Ai Peng New, Edward Samuel Rankin, and Howie Choset. Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, 52:109–142, 2008.

[86] Neil Mathew, Stephen L Smith, and Steven L Waslander. A graph-based approach to multi-robot rendezvous for recharging in persistent tasks. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3497–3502. IEEE, 2013.

[87] Sezgin Kaplan and Ghaith Rabadi. Exact and heuristic algorithms for the aerial refueling parallel machine scheduling problem with due date-to-deadline window and ready times. *Computers & Industrial Engineering*, 62(1):276–285, 2012.

[88] Changjoo Nam and Dylan A Shell. Assignment algorithms for modeling resource contention in multirobot task allocation. *IEEE Transactions on Automation Science and Engineering*, 12(3):889–900, 2015.

[89] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562, 2023.

[90] Emil Juul Jacobsen, Rasmus Greve, and Julian Togelius. Monte mario: platforming with mcts. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, pages 293–300, 2014.

[91] Richard Lorentz. Using evaluation functions in monte-carlo tree search. *Theoretical computer science*, 644:106–113, 2016.

[92] Tobias Graf and Marco Platzner. Adaptive playouts for online learning of policies during monte carlo tree search. *Theoretical Computer Science*, 644:53–62, 2016.

[93] Maciej Swiechowski, Kathryn Merrick, Jacek Mandziuk, and Hussein Abbass. Human-machine cooperation loop in game playing. *Int. J. Adv. Intell. Syst*, 8(3/4):310–323, 2015.

[94] Aijun Bai, Feng Wu, and Xiaoping Chen. Bayesian mixture modelling and inference based thompson sampling in monte-carlo tree search. *Advances in neural information processing systems*, 26, 2013.

[95] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[96] Anji Liu, Jianshu Chen, Mingze Yu, Yu Zhai, Xuewen Zhou, and Ji Liu. Watch the unobserved: A simple approach to parallelizing monte carlo tree search. In *Proceedings of International Conference on Learning Representation (ICLR)*, 2020.

[97] Daniel Beard, Philip Hingston, and Martin Masek. Using monte carlo tree search for replanning in a multistage simultaneous game. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2012.

[98] Steven James, George Konidaris, and Benjamin Rosman. An analysis of monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[99] Nicholas Zerbel and Logan Yliniemi. Multiagent monte carlo tree search. In *Proceedings of the International Conference on Autonomous Agents and MultiAgent Systems*, pages 2309–2311, 2019.

[100] Mohammadreza Daneshvaramoli, Mohammad Sina Kiarostami, Saleh Khalaj Monfared, Helia Karisani, Keivan Dehghannayeri, Dara Rahmati, and Saeid Gorgin. Decentralized communication-less multi-agent task assignment with cooperative monte-carlo tree search. In *Proceedings of International Conference on Control, Automation and Robotics (ICCAR)*, pages 612–616. IEEE, 2020.

[101] Karl Kurzer, Chenyang Zhou, and J Marius Zöllner. Decentralized cooperative planning for automated vehicles with hierarchical monte carlo tree search. In *Proceedings of IEEE intelligent vehicles symposium (IV)*, pages 529–536. IEEE, 2018.

[102] Micah Corah and Nathan Michael. Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots*, 43(2):485–501, 2019.

[103] Chase C. Murray and Amanda G. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technology*, 54:86–109, 2015.

[104] Yaniel Carreno, Ronald P. A. Petrick, and Petillot Yvan. Multi-agent strategy for marine applications via temporal planning. In *Proceedings of IEEE Artificial Intelligence & Knowledge Engineering (AIKE)*, pages 243–250, 2019.

[105] Beth Boardman, Troy Harden, and Sonia Martínez. Limited range spatial load balancing in non-convex environments using sampling-based motion planners. *Autonomous Robots*, pages 1–18, 2018.

[106] Andreas Breitenmoser, Mac Schwager, Jean-Claude Metzger, Roland Siegwart, and Daniela Rus. Voronoi coverage of non-convex environments with a group of networked robots. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4982–4989. IEEE, 2010.

[107] Daniel S. Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[108] Frans A. Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

[109] Mykel J Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

[110] Laëtitia Matignon, Laurent Jeanpierre, and Abdel-Illah Mouaddib. Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26, pages 2017–2023, 2012.

[111] Graeme Best, Oliver M. Cliff, Timothy Patten, Ramgopal R. Mettu, and Robert Fitch. Dec-MCTS: Decentralized planning for multi-robot active perception. *International Journal of Robotics Research*, 38(2-3):316–337, 2019.

[112] Andrew J Smith, Graeme Best, Javier Yu, and Geoffrey A Hollinger. Real-time distributed non-myopic task selection for heterogeneous robotic teams. *Autonomous Robots*, 43:789–811, 2019.

[113] Minglong Li, Wenjing Yang, Zhongxuan Cai, Shaowu Yang, and Ji Wang. Integrating decision sharing with prediction in decentralized planning for multi-agent coordination under uncertainty. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, pages 450–456, 2019.

[114] Weizhe Chen and Lantao Liu. Pareto monte carlo tree search for multi-objective informative planning. In *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[115] Akash Arora, P Michael Furlong, Robert Fitch, Salah Sukkarieh, and Terrence Fong. Multimodal active perception for information gathering in science missions. *Autonomous Robots*, 43:1827–1853, 2019.

[116] Yaohui Guo and X Jessie Yang. Modeling and predicting trust dynamics in human–robot teaming: A bayesian inference approach. *International Journal of Social Robotics*, 13(8): 1899–1909, 2021.

[117] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[118] Zs Lendek, R Babuška, and B De Schutter. Distributed kalman filtering for multiagent systems. In *2007 European Control Conference (ECC)*, pages 2193–2200. IEEE, 2007.

[119] Petar M Djuric, Jayesh H Kotecha, Jianqui Zhang, Yufei Huang, Tadesse Ghirmai, Mónica F Bugallo, and Joaquin Miguez. Particle filtering. *IEEE signal processing magazine*, 20(5): 19–38, 2003.

[120] Ruben Martinez-Cantin. Bayesian optimization with adaptive kernels for robot control. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3350–3356. IEEE, 2017.

[121] Felix Berkenkamp, Andreas Krause, and Angela P Schoellig. Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics. *Machine Learning*, 112 (10):3713–3747, 2023.

[122] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32: 289–353, 2008.

[123] Alessia Benevento, María Santos, Giuseppe Notarstefano, Kamran Paynabar, Matthieu Bloch, and Magnus Egerstedt. Multi-robot coordination for estimation and coverage of unknown spatial fields. In *2020 ieee international conference on robotics and automation (icra)*, pages 7740–7746. IEEE, 2020.

[124] David Portugal and Rui P Rocha. Cooperative multi-robot patrol with bayesian learning. *Autonomous Robots*, 40(5):929–953, 2016.

[125] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25, 2012.

[126] Kun Dong, David Eriksson, Hannes Nickisch, David Bindel, and Andrew G Wilson. Scalable log determinants for gaussian process kernel learning. *Advances in Neural Information Processing Systems*, 30, 2017.

[127] Aaron Ma, Michael Ouimet, and Jorge Cortés. Exploiting bias for cooperative planning in multi-agent tree search. *IEEE Robotics and Automation Letters*, 5(2):1819–1826, 2020.

[128] Haluk Bayram and H Işıl Bozma. Coalition formation games for dynamic multirobot tasks. *The International Journal of Robotics Research*, 35(5):514–527, 2016.

[129] Junyan Hu, Hanlin Niu, Joaquin Carrasco, Barry Lennox, and Farshad Arvin. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Transactions on Vehicular Technology*, 69(12):14413–14423, 2020.

[130] Wei Xi, Xiaobo Tan, and John S Baras. Gibbs sampler-based coordination of autonomous swarms. *Automatica*, 42(7):1107–1119, 2006.

[131] L Pinedo Michael. *Scheduling: theory, algorithms, and systems*. Springer, 2022.

[132] Ali Allahverdi, Chi To Ng, TC Edwin Cheng, and Mikhail Y Kovalyov. A survey of scheduling problems with setup times or costs. *European journal of operational research*, 187(3): 985–1032, 2008.

[133] Ali Allahverdi. A survey of scheduling problems with no-wait in process. *European Journal of Operational Research*, 255(3):665–686, 2016. ISSN 0377-2217. doi: https://doi.org/10.1016/j.ejor.2016.05.036.

[134] Joseph YT Leung. *Handbook of scheduling: algorithms, models, and performance analysis*. CRC press, 2004.

[135] Bo Chen, Chris N Potts, and Gerhard J Woeginger. A review of machine scheduling: Complexity, algorithms and approximability. *Handbook of Combinatorial Optimization: Volume1–3*, pages 1493–1641, 1998.

[136] J Mark Keil. On the complexity of scheduling tasks with discrete starting times. *Operations research letters*, 12(5):293–295, 1992.

[137] Roberta De Santis, Roberto Montanari, Giuseppe Vignali, and Eleonora Bottani. An adapted ant colony optimization algorithm for the minimization of the travel distance of pickers in manual warehouses. *European Journal of Operational Research*, 267(1):120–137, 2018.

[138] Jelmer Pier van der Gaast, Bolor Jargalsaikhan, and Kees Jan Roodbergen. Dynamic batching for order picking in warehouses. In *Proceedings of International Material Handling Research Colloquium (IMHRC)*. Digital Commons Georgia Southern, 2018.

[139] Xiaoyong Tang, Xiaochun Li, and Zhuojun Fu. Budget-constraint stochastic task scheduling on heterogeneous cloud systems. *Concurrency and Computation: Practice and Experience*, 29(19):e4210, 2017.

[140] Yifan Zhou, Jindan Miao, Bin Yan, and Zhisheng Zhang. Stochastic resource-constrained project scheduling problem with time varying weather conditions and an improved estimation of distribution algorithm. *Computers & Industrial Engineering*, 157:107322, 2021.

[141] Jan Böttcher, Andreas Drexl, Rainer Kolisch, and Frank Salewski. Project scheduling under partially renewable resource constraints. *Management Science*, 45(4):543–559, 1999.

[142] Hannah Bakker, Fabian Dunke, and Stefan Nickel. A structuring review on multi-stage optimization under uncertainty: Aligning concepts from theory and practice. *Omega*, 96: 102080, 2020.

[143] Kai Watermeyer and Jürgen Zimmermann. A branch-and-bound procedure for the resource-constrained project scheduling problem with partially renewable resources and general temporal constraints. *OR spectrum*, 42:427–460, 2020.

[144] Cheng-Hsiang Liu. Mathematical programming formulations for single-machine scheduling problems while considering renewable energy uncertainty. *International Journal of Production Research*, 54(4):1122–1133, 2016.

[145] Wencong Su, Jianhui Wang, and Jaehyung Roh. Stochastic energy scheduling in microgrids with intermittent renewable energy resources. *IEEE Transactions on Smart grid*, 5(4):1876–1883, 2013.

[146] Manijeh Alipour, Behnam Mohammadi-Ivatloo, and Kazem Zare. Stochastic risk-constrained short-term scheduling of industrial cogeneration systems in the presence of demand response programs. *Applied Energy*, 136:393–404, 2014.

[147] Ye Ren, Ponnuthurai Nagaratnam Suganthan, and Narasimalu Srikanth. A comparative study of empirical mode decomposition-based short-term wind speed forecasting methods. *IEEE Transactions on Sustainable Energy*, 6(1):236–244, 2014.

[148] Yongli Ji, Qingshan Xu, Jun Zhao, Yongbiao Yang, and Lu Sun. Day-ahead and intra-day optimization for energy and reserve scheduling under wind uncertainty and generation outages. *Electric Power Systems Research*, 195:107133, 2021.

[149] Yi Wang, Dawei Qiu, and Goran Strbac. Multi-agent deep reinforcement learning for resilience-driven routing and scheduling of mobile energy storage systems. *Applied Energy*, 310:118575, 2022.

[150] Anubhav Kumar Pandey, Vinay Kumar Jadoun, and Jayalakshmi N. Sabhahit. Real-time peak valley pricing based multi-objective optimal scheduling of a virtual power plant considering renewable resources. *Energies*, 2022.

[151] S Surender Reddy, PR Bijwe, and Abhijit R Abhyankar. Real-time economic dispatch considering renewable power generation variability and uncertainty over scheduling period. *IEEE Systems journal*, 9(4):1440–1451, 2014.

[152] Giorgio Cau, Daniele Cocco, Mario Petrollese, Søren Knudsen Kær, and Christian Milan. Energy management strategy based on short-term generation scheduling for a renewable microgrid using a hydrogen storage system. *Energy Conversion and Management*, 87:820–831, 2014.

[153] Iver Bakken Sperstad and Magnus Korpås. Energy storage scheduling in distribution systems considering wind and photovoltaic generation uncertainties. *Energies*, 12(7):1231, 2019.

[154] Xiong Wu, Shixiong Qi, Zhao Wang, Chao Duan, Xiuli Wang, and Furong Li. Optimal scheduling for microgrids with hydrogen fueling stations considering uncertainty using data-driven approach. *Applied energy*, 253:113568, 2019.

[155] Weifeng Liu, Feilin Zhu, Tongtiegang Zhao, Hao Wang, Xiaohui Lei, Ping-an Zhong, and Vasilis Fthenakis. Optimal stochastic scheduling of hydropower-based compensation for combined wind and photovoltaic power outputs. *Applied Energy*, 276:115501, 2020.

[156] John A Stankovic, Krithivasan Ramamritham, and Shengchang Cheng. Evaluation of a flexible task scheduling algorithm for distributed hard real-time systems. *IEEE Transactions on computers*, 100(12):1130–1143, 1985.

[157] John Regehr. Scheduling tasks with mixed preemption relations for robustness to timing faults. In *Proceedings of IEEE Real-Time Systems Symposium (RTSS)*, pages 315–326. IEEE, 2002.

[158] Sajib K Biswas, Amit Rauniyar, and Pranab K Muhuri. Multi-objective bayesian optimization algorithm for real-time task scheduling on heterogeneous multiprocessors. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2844–2851. IEEE, 2016.

[159] David G. Kendall. Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. *The Annals of Mathematical Statistics*, 24(3):338 – 354, 1953.

[160] A Ravi Ravindran. *Operations research and management science handbook*. CRC press, 2016.

[161] Angélica Munoz-Meléndez, Prithviraj Dasgupta, and William Lenagh. A stochastic queueing model for multi-robot task allocation. In *ICINCO (1)*, pages 256–261, 2012.

[162] Asefeh Hasani Goodarzi, Eleen Diabat, Armin Jabbarzadeh, and Marc Paquet. An m/m/c queue model for vehicle routing problem in multi-door cross-docking environments. *Computers & Operations Research*, 138:105513, 2022.

[163] Buddhadeb Pradhan, Veena Goswami, Rabindra K Barik, and Sudipta Sahana. An integrated strategy-based game-theoretic model and decentralized queueing system for mobile multi-robot task coordination. *Decision Analytics Journal*, 7:100254, 2023.

[164] Yizhou Huang, Hamza Dugmag, Timothy D Barfoot, and Florian Shkurti. Stochastic planning for asv navigation using satellite images. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1055–1061. IEEE, 2023.

[165] Theodore Tsiligirides. Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35:797–809, 1984.

[166] Vicente Campos, Rafael Martí, Jesús Sánchez-Oro, and Abraham Duarte. Grasp with path relinking for the orienteering problem. *Journal of the Operational Research Society*, 65 (12):1800–1813, 2014.

[167] Gorka Kobeaga, María Merino, and Jose A Lozano. An efficient evolutionary algorithm for the orienteering problem. *Computers & Operations Research*, 90:42–59, 2018.

[168] Cédric Verbeeck, Kenneth Sörensen, E-H Aghezzaf, and Pieter Vansteenwegen. A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, 236(2):419–432, 2014.

[169] Steven E Butt and Tom M Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21(1):101–111, 1994.

[170] I-Ming Chao, Bruce L Golden, and Edward A Wasil. The team orienteering problem. *European journal of operational research*, 88(3):464–474, 1996.

[171] Hao Tang and Elise Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6):1379–1407, 2005.

[172] Claudia Archetti, Dominique Feillet, Alain Hertz, and Maria Grazia Speranza. The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society*, 60:831–842, 2009.

[173] Sylvain Boussier, Dominique Feillet, and Michel Gendreau. An exact algorithm for team orienteering problems. *4OR*, 5:211–230, 2007.

[174] Ann M Campbell, Michel Gendreau, and Barrett W Thomas. The orienteering problem with stochastic travel and service times. *Annals of Operations Research*, 186(1):61–81, 2011.

[175] Irina Dolinskaya, Zhenyu Edwin Shi, and Karen Smilowitz. Adaptive orienteering problem with stochastic travel times. *Transportation Research Part E: Logistics and Transportation Review*, 109:1–19, 2018.

[176] Bruce L. Golden, Subramanian Raghavan, and Edward A. Wasil. *The vehicle routing problem: latest advances and new challenges*, volume 43. Springer Science & Business Media, 2008.

[177] George B Dantzig and John H Ramser. The truck dispatching problem. *Management science*, 6(1):80–91, 1959.

[178] Dimitris J Bertsimas and Garrett Van Ryzin. Stochastic and dynamic vehicle routing in the euclidean plane with multiple capacitated vehicles. *Operations Research*, 41(1):60–76, 1993.

[179] Pasquale Grippa, Doris A Behrens, Friederike Wall, and Christian Bettstetter. Drone delivery systems: job assignment and dimensioning. *Autonomous Robots*, 43(2):261–274, 2019.

[180] Zhiwei Yang, Jan-Paul van Osta, Barry van Veen, Rick van Krevelen, Richard van Klaveren, Andries Stam, Joost Kok, Thomas Bäck, and Michael Emmerich. Dynamic vehicle routing with time windows in theory and practice. *Natural Computing*, 16(1):119–134, 2017.

[181] Marco Pavone, Alessandro Arsie, Emilio Frazzoli, and Francesco Bullo. Equitable partitioning policies for robotic networks. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 2356–2361. IEEE, 2009.

[182] Guy Desaulniers, Jacques Desrosiers, Andreas Erdmann, Marius M. Solomon, and François Soumis. *VRP with Pickup and Delivery*, pages 225–242. SIAM, 2002.

[183] Christopher C. Sotzing and David M. Lane. Improving the coordination efficiency of limited-communication multi–autonomus underwater vehicle operations using a multiagent architecture. *Journal of Field Robotics*, 35(3):312–329, 2018.

[184] P. Mojiri Forooshani and M. Jenkin. Sensor coverage with a heterogeneous fleet of autonomous surface vessels. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

[185] Tapovan Lolla, Pierre FJ Lermusiaux, Mattheus P Ueckermann, and Patrick J Haley. Time-optimal path planning in dynamic flows using level set equations: theory and schemes. *Ocean Dynamics*, 64:1373–1397, 2014.

[186] Alberto Alvarez, Andrea Caiti, and Reiner Onken. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering*, 29(2):418–429, 2004.

[187] Mohan Krishna Nutalapati, Shruti Joshi, and Ketan Rajawat. Online utility-optimal trajectory design for time-varying ocean environments. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6853–6859. IEEE, 2019.

[188] James Ju Heon Lee, Chanyeol Yoo, Raewyn Hall, Stuart Anstee, and Robert Fitch. Energy-optimal kinodynamic planning for underwater gliders in flow fields. In *Proceedings of Australasian Conference of Robotics and Automation (ACRA)*, 2017.

[189] Chanyeol Yoo, Stuart Anstee, and Robert Fitch. Stochastic path planning for autonomous underwater gliders with safety constraints. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3725–3732, 2019.

[190] Chanyeol Yoo, James Ju Heon Lee, Stuart Anstee, and Robert Fitch. Path planning in uncertain ocean currents using ensemble forecasts. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[191] Sonia Hernandez and Derek A Paley. Three-dimensional motion coordination in a time-invariant flowfield. In *Proceedings of the 48h IEEE Conference on Decision and Control*

(CDC) held jointly with 2009 28th Chinese Control Conference, pages 7043–7048. IEEE, 2009.

[192] Ariella Mansfield, Douglas G Macharet, and M Ani Hsieh. Energy-efficient orienteering problem in the presence of ocean currents. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10081–10087. IEEE, 2022.

[193] Ariella Mansfield, Douglas G Macharet, and M Ani Hsieh. Energy-efficient team orienteering problem in the presence of time-varying ocean currents. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 838–843. IEEE, 2023.

[194] Ehsan Ardjmand, Heman Shakeri, Manjeet Singh, and Omid Sanei Bajgiran. Minimizing order picking makespan with multiple pickers in a wave picking warehouse. *International Journal of Production Economics*, 206:169–183, 2018.

[195] İbrahim Muter and Temel Öncan. Order batching and picker scheduling in warehouse order picking. *IISE Transactions*, 54(5):435–447, 2022.

[196] Lijie Zhou, Chengran Lin, Qian Ma, and Zhengcai Cao. A learning-based iterated local search algorithm for order batching and sequencing problems. In *Proceedings of IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1741–1746. IEEE, 2022.

[197] Daniel Schubert, André Scholz, and Gerhard Wäscher. Integrated order picking and vehicle routing with due dates. *OR Spectrum*, 40(4):1109–1139, 2018.

[198] André Scholz, Daniel Schubert, and Gerhard Wäscher. Order picking with multiple pickers and due dates–simultaneous solution of order batching, batch assignment and sequencing, and picker routing problems. *European Journal of Operational Research*, 263(2):461–478, 2017.

[199] Vaggelis Giannikas, Wenrong Lu, Brian Robertson, and Duncan McFarlane. An interventionist strategy for warehouse order picking: Evidence from two case studies. *International Journal of Production Economics*, 189:63–76, 2017.

[200] Jose IU Rubrico, Toshimitu Higashi, Hirofumi Tamura, and Jun Ota. Online rescheduling of multiple picking agents for warehouse management. *Robotics and Computer-Integrated Manufacturing*, 27(1):62–71, 2011.

[201] Igor Halperin. *Reinforcement Learning and Stochastic Optimization: A Unified Framework for Sequential Decisions: by Warren B. Powell (ed.), Wiley (2022). Hardback. ISBN 9781119815051.*, volume 22. Taylor & Francis, 2022.

[202] Moloud Abdar, Farhad Pourpanah, Sadiq Hussain, Dana Rezazadegan, Li Liu, Mohammad Ghavamzadeh, Paul Fieguth, Xiaochun Cao, Abbas Khosravi, U Rajendra Acharya, et al. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information fusion*, 76:243–297, 2021.

[203] Mario Di Nardo, Mariano Clericuzio, Teresa Murino, and Chiara Sepe. An economic order quantity stochastic dynamic optimization model in a logistic 4.0 environment. *Sustainability*, 12(10):4075, 2020.

[204] Hector Flores and J Rene Villalobos. A stochastic planning framework for the discovery of complementary, agricultural systems. *European Journal of Operational Research*, 280(2): 707–729, 2020.

[205] Marco Tranzatto, Takahiro Miki, Mihir Dharmadhikari, Lukas Bernreiter, Mihir Kulkarni, Frank Mascarich, Olov Andersson, Shehryar Khattak, Marco Hutter, Roland Siegwart, et al. Cerberus in the darpa subterranean challenge. *Science Robotics*, 7(66), 2022.

[206] David Silver and Joel Veness. Monte-carlo planning in large pomdps. *Advances in neural information processing systems*, 23, 2010.

[207] Timothy Patten, Wolfram Martens, and Robert Fitch. Monte carlo planning for active object classification. *Autonomous Robots*, 42(2):391–421, 2018.

[208] Stefano Carpin, Yin-Lam Chow, and Marco Pavone. Risk aversion in finite markov decision processes using total cost criteria and average value at risk. In *Proceedings of IEEE international conference on robotics and automation (ICRA)*, pages 335–342. IEEE, 2016.

[209] Carlo Filippi, Gianfranco Guastaroba, and Maria Grazia Speranza. Conditional value-at-risk beyond finance: a survey. *International Transactions in Operational Research*, 27(3): 1277–1319, 2020.

[210] Graeme Best, Michael Forrai, Ramgopal R Mettu, and Robert Fitch. Planning-aware communication for decentralised multi-robot coordination. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1050–1057. IEEE, 2018.

[211] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:566–580, 1996.

[212] Kim Reh, Linda Spilker, Jonathan I Lunine, J Hunter Waite, Morgan L Cable, Frank Postberg, and Karla Clark. Enceladus life finder: The search for life in a habitable moon. In *Proceedings of IEEE Aerospace Conference*, pages 1–8. IEEE, 2016.

[213] Wikipedia, the free encyclopedia. Miyamoto crater, 2013. URL `https://mars.nasa.gov/msl/news/images/20080918a/Miyamoto_ellipse2.jpg`. [Online; accessed November 07, 2019].

[214] Jennifer Dooley. Mission concept for a europa lander. In *2018 IEEE Aerospace Conference*, pages 1–10. IEEE, 2018.

[215] Patrick A. Plonski, Joshua Vander Hook, Cheng Peng, Narges Noori, and Volkan Isler. Environment exploration in sensing automation for habitat monitoring. *Transactions on Automation Science and Engineering*, 14(1):25–38, 2016.

[216] Daniel L. Bongiorno, Mitch Bryson, Tom C. L. Bridge, Donald G. Dansereau, and Stefan B. Williams. Conregistered hyperspectral and stereo image seafloor mapping from an autonomous underwater vechicle. *Journal of Field Robotics*, 35(3):312–329, 2018.

[217] Jackson Shields, Oscar Pizarro, and Stefan B. Williams. Towards adapive benthic habitat mapping. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 9263–9270, 2020.

[218] Daniel L Rudnick, Russ E Davis, Charles C Eriksen, David M Fratantoni, and Mary Jane Perry. Underwater gliders for ocean research. *Marine Technology Society Journal*, 38(2): 73–84, 2004.

[219] Mark Grasmueck, Gregor P. Eberli, David A. Viggiano, and Thiago Correa. Autonomous underwater vehicle (AUV) mapping reveals coral mound distribution, morphology, and oceanography in deep water of the Straits of Florida. *Geophysical Research Letters*, 33 (23):L23616, 2006.

[220] Sergio Jaramillo and Geno Pawlak. AUV-based bed roughness mapping over a tropical reef. *Coral Reefs*, 30(1):11–23, 2011.

[221] Stefan B. Williams, Oscar R. Pizarro, Michael V. Jakuba, Craig R. Johnson, Neville S. Barrett, Russell C. Babcock, Gary A. Kendrick, Peter D. Steinberg, Andrew J. Heyward, Peter J. Doherty, Ian Mahon, Matthew Johnson-Roberson, Daniel Steinberg, and Ariell Friedman. Monitoring of benthic reference sites. *IEEE Robotics & Automation Magazine*, 19(1):73–84, 2012.

[222] F. Caratori Tontini, C. E. J. de Ronde, J. C. Kinsey, A. Soule, D. Yoerger, and L. Cocchi. Geophysical modelling of collapse-prone zones at rumble III seamount, southern Pacific Ocean, New Zealand. *Geochemistry, Geophysics, Geosystems*, 14(10):4667–4680, 2013.

[223] David A. Clague, Brian M. Dreyer, Jennifer B. Paduan, Julie F. Martin, David W. Caress, James B. Gill, Deborah S. Kelley, Hans Thomas, Ryan A. Portner, John R. Delaney, Thomas P. Guilderson, and Mary L. McGann. Eruptive and techtonic history of the Endeavour Segment, Juan de Fuca Ridge, based on AUV mapping data and lava flow ages. *Geochemistry, Geophysics, Geosystems*, 15(8):3364–3391, 2014.

[224] Jannifer B. Paduan, David A. Clague, David W. Caress, and Hans Thomas. High-resolution AUV mapping and ROV sampling of mid-ocean ridges. In *Proceedings of MTS/IEEE OCEANS*, pages 1–8, 2016.

[225] Stephanie M. Petillo and Henrik Schmidt. Autonomous and adaptive underwater plume detection and tracking with AUVs: Concepts, methods, and available technology. *Proceedings of International Federation of Automatic Control (IFAC)*, 45(27):232–237, 2012.

[226] Yoshiaki Maeda, Kiminori Shitashima, and Atsushi Sakamoto. Mapping observations using auv and numerical simulations of leaked CO2 diffusion in sub-seabed CO2 release experiment at ardmucknish bay. *International Journal of Greenhouse Gas Control*, 38:143–152, 2015.

[227] Ki Myung Brian Lee, James Ju Heon Lee, Chanyeol Yoo, Ben Hollings, and Robert Fitch. Active perception for plume source localisation with underwater gliders. In *Proceedings of Australasian Conference of Robotics and Automation (ACRA)*, 2018.

[228] Chris Roman, Gabrielle Inglis, and Bryan McGilvray. Lagrangian floats as sea floor imaging platforms. *Continential Shelf Reseach*, 31(15):1592–1598, 2011.

[229] Eemeli Aro. The utility of an autonomous multi-robot system of underwater floats. *Proceedings of International Federation of Automatic Control (IFAC)*, 45(28):55–59, 2012.

[230] Michael P. Meredith, Oscar Schofield, Louise Newman, Ed Urban, and Michael Sparrow. The vision for a southern ocean observing system. *Current Opinion in Environmental Sustainability*, 5(3):306–313, 2013.

[231] Enrique Fernández-Perdomo, Jorge Cabrera-Gámez, Daniel Hernández-Sosa, Josep Isern-González, Antonio C Domínguez-Brito, Alex Redondo, Josep Coca, Antonio G Ramos, Enrique Álvarez Fanjul, and Marcos García. Path planning for gliders using regional ocean models: Application of pinzón path planner with the eseoat model and the ru27 trans-atlantic flight data. In *Proceedings of IEEE OCEANS*, pages 1–10. IEEE, 2010.

[232] Enrique Fernández-Perdomo, Daniel Hernández-Sosa, Josep Isern-González, Jorge Cabrera-Gámez, Antonio C Dominguez-Brito, and Víctor Prieto-Marañón. Single and multiple glider path planning using an optimization-based approach. In *Proceedings of IEEE OCEANS*, pages 1–10, 2011.

[233] Chien-Chou Shih, Mong-Fong Horng, Tien-Szu Pan, Jeng-Shyang Pan, and Chun-Yu Chen. A genetic-based effective approach to path-planning of autonomous underwater glider with upstream-current avoidance in variable oceans. *Soft Computing*, 21(18):5369–5386, 2017.

[234] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[235] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of European Conference on Machine Learning (ECML)*, pages 282–293. Springer, 2006.

[236] K. Y. Cadmus To, Ki Myung Brian Lee, Chanyeol Yoo, Stuart Anstee, and Robert Fitch. Streamlines for motion planning in underwater currents. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4619–4625, 2019.

[237] K. Y. Cadmus To, Chanyeol Yoo, Stuart Anstee, and Robert Fitch. Distance functions and steering heuristics for streamline-based planning in flow fields. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1867–1873, 2020.

[238] Ki Myung Brian Lee, Chanyeol Yoo, Ben Hollings, Stuart Anstee, Shoudong Huang, and Robert Fitch. Online estimation of ocean current from sparse GPS data for underwater vehicles. *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2019.

[239] Mehdi Lamiri, Xiaolan Xie, Alexandre Dolgui, and Frédéric Grimaud. A stochastic model for operating room planning with elective and emergency demand for surgery. *European Journal of Operational Research*, 185(3):1026–1037, 2008.

[240] TC Poon, King Lun Choy, Felix TS Chan, and Henry CW Lau. A real-time production operations decision support system for solving stochastic production material demand problems. *Expert Systems with Applications*, 38(5):4829–4838, 2011.

[241] Aliaa Alnaggar, Fatma Gzara, and James H Bookbinder. Distribution planning with random demand and recourse in a transshipment network. *EURO Journal on Transportation and Logistics*, 9(1):100007, 2020.

[242] Vivek Khanzode and Bhavin Shah. A comprehensive review of warehouse operational issues. *International Journal of Logistics Systems and Management*, 26:346, 01 2017. doi: 10.1504/IJLSM.2017.10002597.

[243] René M. B. M. de Koster, Tho Le-Duc, and Kees Jan Roodbergen. Design and control of warehouse order picking: A literature review. *European Journal of Operational Research*, 182:481–501, 2007.

[244] Michael R Garey and David S Johnson. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

[245] Franco P Preparata and Michael I Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012.

[246] David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. Technical report, Stanford, 2006.

[247] Stuart Lloyd. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[248] Franz Aurenhammer. Power diagrams: properties, algorithms and applications. *SIAM Journal on Computing*, 16(1):78–96, 1987.

[249] Mengfei Yu and René BM De Koster. The impact of order batching and picking area zoning on order picking system performance. *European Journal of Operational Research*, 198(2): 480–490, 2009.

[250] René de Koster. Performance approximation of pick-to-belt orderpicking systems. *European Journal of Operational Research*, 72(3):558–573, 1994.

[251] Fangyu Chen, Yongchang Wei, and Hongwei Wang. A heuristic based batching and assigning method for online customer orders. *Flexible Services and Manufacturing Journal*, 30 (4):640–685, 2018.

[252] JFC Kingman. The single server queue in heavy traffic. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 57, pages 902–904. Cambridge University Press, 1961.

[253] Haiping Xu. *Optimal policies for stochastic and dynamic vehicle routing problems*. PhD thesis, Massachusetts Institute of Technology, 1994.

[254] H. Ratliff and Arnon Rosenthal. Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem. *Operations Research*, 31:507–521, 06 1983. doi: 10.1287/opre.31.3.507.

[255] Sanjeev Arora. Polynomial time approximation schemes for euclidean tsp and other geometric problems. In *Proceedings of Conference on Foundations of Computer Science*, pages 2–11. IEEE, 1996.

[256] Ruben D'Haen, Kris Braekers, and Katrien Ramaekers. Integrated scheduling of order picking operations under dynamic order arrivals. *International Journal of Production Research*, 61(10):3205–3226, 2023.

[257] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International journal of robotics research*, 34(7):883–921, 2015.

[258] Zhen Wang, Qiaoyan Bian, Huanhai Xin, and Deqiang Gan. A distributionally robust coordinated reserve scheduling model considering cvar-based wind power reserve requirements. *IEEE Transactions on Sustainable Energy*, 7(2):625–636, 2015.

[259] Ran Wang, Ping Wang, and Gaoxi Xiao. A robust optimization approach for energy generation scheduling in microgrids. *Energy Conversion and Management*, 106:597–607, 2015.

[260] Michelangelo Ceci, Roberto Corizzo, Donato Malerba, and Aleksandra Rashkovska. Spatial autocorrelation and entropy for renewable energy forecasting. *Data Mining and Knowledge Discovery*, 33(3):698–729, 2019.

[261] Shahram Hanifi, Xiaolei Liu, Zi Lin, and Saeid Lotfian. A critical review of wind power forecasting methods—past, present and future. *Energies*, 13(15):3764, 2020.

[262] Wenqing Xu, Like Ning, and Yong Luo. Wind speed forecast based on post-processing of numerical weather predictions using a gradient boosting decision tree algorithm. *Atmosphere*, 11(7):738, 2020.

[263] Ren Cai, Sen Xie, Bozhong Wang, Ruijiang Yang, Daosen Xu, and Yang He. Wind speed forecasting based on extreme gradient boosting. *IEEE Access*, 8:175063–175069, 2020.

[264] Richard Swinbank, Masayuki Kyouda, Piers Buchanan, Lizzie Froude, Thomas M Hamill, Tim D Hewson, Julia H Keller, Mio Matsueda, John Methven, Florian Pappenberger, et al. The tigge project and its achievements. *Bulletin of the American Meteorological Society*, 97(1):49–67, 2016.

[265] Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, et al. The era5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020.

Appendices

A Supplementary algorithmic and implementation details

A.1 Float dynamic model

A proper evaluation of the float and vessel path quality in flow field environment is needed for valid deployment planning. One approach to path evaluation is to numerically integrate the vehicle dynamic model across the flow field. However, deriving for a full dynamic model is complex and expensive from a planning perspective. A simplified approach using kinematic models with directly controllable velocity provides a quick approximation to the vehicle path. In this appendix, we define the flow field model used for the simulations and the float and vessel model. We also define the vehicle path trajectory given the initial conditions and the control input.

A.1.1 Flow field model

Given a position $X = [x, y, z]^T$, the Taylor-Green model is used to represent the oceanic flow field. For max flow field strength A and eddy width s:

$$V_{flow,Taylor}(\gamma_k, \delta_k) = \begin{bmatrix} -A\sin(\pi x/s)\cos(\pi y/s) \\ A\cos(\pi x/s)\sin(\pi y/s) \\ 0 \end{bmatrix}. \tag{A.1}$$

A common practice in modelling oceanic flow field is to assume the flow field in the z-axis is zero. In practice, there does exist a flow in the z-axis, but it is negligible compared to the other vector component. The flow field model can be superimposed with other flow field to create interesting streamlines. Suppose we had a linear flow field that rotates its direction with a period of T:

$$V_{flow}(X,t) = V_{flow,Taylor}(X,t) + \begin{bmatrix} \cos(2\pi t/T) \\ \sin(2\pi t/T) \\ 0 \end{bmatrix} \tag{A.2}$$

A.1.2 Float dynamics

The float does not possess forward propulsion and is required to drift along the current to relocate. The float can however move along the depth-axis. Suppose the diving/surfacing velocity of the float is 0.5m/s. Assuming a kinematic model where the velocity is directly controllable, the float dynamic in still water is expressed as:

$$V_{float} = \begin{bmatrix} u_{float} \\ v_{float} \\ w_{float} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \pm 0.5 \end{bmatrix} \tag{A.3}$$

By considering drift into the float dynamic, the net float velocity under the influence of flow field is expressed as:

$$V_{total,float}(X,t) = V_{float} + V_{flow}(X,t) = \begin{bmatrix} u_{flow}(X,t) \\ v_{flow}(X,t) \\ \pm 0.5 \end{bmatrix} \tag{A.4}$$

A.1.3 Vessel dynamics

Similar to the float dynamics, we model the dynamics of the surface vessel as a kinematic model with directly controllable velocity. Assuming a constant speed V_s with heading θ and instantaneous turning rate:

$$V_{vessel} = \begin{bmatrix} u_{vessel} \\ v_{vessel} \\ w_{vessel} \end{bmatrix} = V_s \begin{bmatrix} \cos\theta(t) \\ \sin\theta(t) \\ 0 \end{bmatrix} \tag{A.5}$$

The vessel dynamics under the flow field influence is expressed as:

$$V_{total,vessel}(X,t) = V_{vessel} + V_{flow}(X,t) = \begin{bmatrix} V_s \cos\theta(t) + u_{flow}(X,t) \\ V_s \sin\theta(t) + v_{flow}(X,t) \\ 0 \end{bmatrix} \tag{A.6}$$

A.1.4 Trajectories

The trajectory is evaluated through forward integration. To avoid using extremely small time-steps to derive the trajectory, we use the two-step Adams-Bashforth method:

$$\begin{aligned} X_n &= X_{n-1} + \left(1.5V_{total}\left(X_{n-1}, t_{n-1}\right) - 0.5V_{total}\left(X_{n-2}, t_{n-2}\right)\right) \cdot dt \\ t_n &= n \cdot dt + t_0 \end{aligned} \tag{A.7}$$

Where:

- $X_0 = [x_0, y_0, z_0]^T$ is the initial start position
- t_0 is the initial start time
- $X_1 = X_0 + V_{total}(X_0, t_0) \cdot dt$