Multi-robot mission optimisation: An online approach for optimised, long range inspection and sampling missions

Nikolaos Tsiogkas

Ocean Systems Laboratory

School of Engineering and Physical Sciences

Heriot-Watt University

A thesis submitted for the degree of

Doctor of Philosophy

May 5, 2019

Abstract

Mission execution optimisation is an essential aspect for the real world deployment of robotic systems. Execution optimisation can affect the outcome of a mission by allowing longer missions to be executed or by minimising the execution time of a mission.

This work proposes methods for optimising inspection and sensing missions undertaken by a team of robots operating under communication and budget constraints. Regarding the inspection missions, it proposes the use of an information sharing architecture that is tolerant of communication errors combined with multi-robot task allocation approaches that are inspired by the optimisation literature. Regarding the optimisation of sensing missions under budget constraints novel heuristic approaches are proposed that allow optimisation to be performed online. These methods are then combined to allow the online optimisation of long-range sensing missions performed by a team of robots communicating through a noisy channel and having budget constraints.

All the proposed approaches have been evaluated using simulations and real-world robots. The gathered results are discussed in detail and show the benefits and the constraints of the proposed approaches, along with suggestions for further future directions.

To my parents, Dimitris and Stella, and my partner Dragana.

Acknowledgements

It has been already six years since I have joined the Ocean Systems Laboratory (OSL) under the supervision of Prof. David Lane and Prof. Yvan Petillot. From the first day the team at the OSL welcomed me and made me feel it as a second family, supporting me to grow and helping me with any issues that were presented during the PhD process.

I would specifically like to thank Prof. David Lane who gave me the opportunity to study as PhD in the OSL and for allowing me to become an independent and confident researcher as he said in my first days in the lab. In addition I would like to thank Dr. Zeyn Saigol whose invaluable guidance in the beginning of my PhD allowed me to have a solid understanding of how research should be performed. I would also like to thank my family and friends, both in the OSL and allover the world, that were always there to support me and discuss any ideas and issues. Special thanks should be attributed to my partner Dragana, who patiently supported me through the last and hardest, yet most productive and enjoyable parts of the PhD. Finally, I would like to thank all the people at Intermodalics who supported and facilitated me finishing this work. To all of you who are mentioned, and to those who may be forgotten by mistake you will have my eternal gratitude. Thank you.

ACADEMIC REGISTRY

Research Thesis Submission

Please note this form should be bound into the submitted thesis.

Name:	*Nikolaos Tsiogkas*		
School:	EPS		
Version: *(i.e. First, Resubmission, Final)*	Final	Degree Sought:	*PhD*

Declaration

In accordance with the appropriate regulations I hereby submit my thesis and I declare that:

1. The thesis embodies the results of my own work and has been composed by myself
2. Where appropriate, I have made acknowledgement of the work of others
3. Where the thesis contains published outputs under Regulation 6 (9.1.2) these are accompanied by a critical review which accurately describes my contribution to the research and, for multi-author outputs, a signed declaration indicating the contribution of each author (complete Inclusion of Published Works Form – see below)
4. The thesis is the correct version for submission and is the same version as any electronic versions submitted*.
5. My thesis for the award referred to, deposited in the Heriot-Watt University Library, should be made available for loan or photocopying and be available via the Institutional Repository, subject to such conditions as the Librarian may require
6. I understand that as a student of the University I am required to abide by the Regulations of the University and to conform to its discipline.
7. Inclusion of published outputs under Regulation 6 (9.1.2) shall not constitute plagiarism.
8. I confirm that the thesis has been verified against plagiarism via an approved plagiarism detection application e.g. Turnitin.

* *Please note that it is the responsibility of the candidate to ensure that the correct version of the thesis is submitted.*

Signature of Candidate:		Date:	11/07/2019

Submission

Submitted By *(name in capitals):*	
Signature of Individual Submitting:	
Date Submitted:	

For Completion in the Student Service Centre (SSC)

Received in the SSC by *(name in capitals):*			
Method of Submission *(Handed in to SSC; posted through internal/external mail):*			
*E-thesis Submitted (**mandatory for final theses**)*			
Signature:		Date:	

ACADEMIC REGISTRY
Inclusion of Published Works

Declaration

This thesis contains one or more multi-author published works. In accordance with Regulation 6 (9.1.2) I hereby declare that the contributions of each author to these publications is as follows:

Citation details	N. Tsiogkas and D. M. Lane, An Evolutionary Algorithm for Online, Resource-Constrained, Multivehicle Sensing Mission Planning, IEEE Robotics and Automation Letters, Volume 3, Number 2, pages 1199 – 1206, 2018
Author 1	Theoretical and algorithmic work, full implementation, experimental results, result analysis, paper writing
Author 2	Supervision
Signature:	
Date:	

Citation details	N. Tsiogkas, V. De Carolis and D. M. Lane, Towards an Online Heuristic Method for Energy-Constrained Underwater Sensing Mission Planning, Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on, 2017
Author 1	Theoretical and algorithmic work, full implementation, experimental results, result analysis, paper writing
Author 2	Result analysis, paper writing
Signature:	
Date:	

Citation details	N. Tsiogkas, V. De Carolis and D. M. Lane, Energy-constrained informative routing for AUVs, OCEANS 2016-Shanghai, 2016
Author 1	Theoretical and algorithmic work, full implementation, experimental results, result analysis, paper writing
Author 2	Result analysis, paper writing
Signature:	
Date:	

Contents

List of Figures

List of Tables

List of Algorithms

Thesis' Publications

During the course of the PhD studies whose outcome is this work the following publications have been made:

N. Tsiogkas, G. Papadimitriou, Z. Saigol, and D. Lane, "Efficient multi-auv cooperation using semantic knowledge representation for underwater archaeology missions", in *Oceans '14 St. John's*, 2014.

N. Tsiogkas, Z. Saigol, and D. Lane, "Distributed multi-auv cooperation methods for underwater archaeology", in *OCEANS 2015-Genova*, IEEE, 2015, pp. 1–5.

N. Tsiogkas, V. De Carolis, and D. M. Lane, "Energy-constrained informative routing for auvs", in *OCEANS 2016-Shanghai*, IEEE, 2016, pp. 1–5.

N. Tsiogkas, V. De Carolis, and D. Lane, "Towards an online heuristic method for energy-constrained underwater sensing mission planning", in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, IEEE, 2017.

N. Tsiogkas and D. M. Lane, "An evolutionary algorithm for online, resource-constrained, multivehicle sensing mission planning", *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1199–1206, Apr. 2018. DOI: 10.1109/LRA.2018.2794578.

Chapter 1

Introduction

The last few years have seen significant advances in robotic technology which have led to reliable and affordable systems, which are becoming an indispensable assistant to science and industry. The use of autonomous robots facilitates the execution of tasks that are usually dangerous, expensive or time-consuming when performed by humans. Some examples include space exploration [1], underwater archaeology [2] and search and rescue [3].

One critical use case of robotics technology is data collection. Robots can be used to facilitate and automate tasks like structure inspection or sample collection. Structure inspection is vital as ageing infrastructure requires labour intensive inspection and maintenance operations. Robots can be used to speed the inspection procedure allowing a more substantial portion of the infrastructure to be inspected and providing more data on where maintenance is required. This, in turn, can reduce the damages caused by accidents due to poor maintenance. Unfortunately, even when all precautions are taken, accidents can still happen. In such cases, robots can be deployed to gather data regarding the damages caused. Sample collection is one of the ways to study the effects of such accidents and try to mitigate them. For example, in cases involving chemical pollution or environmental degradation, information is required regarding the extent of the problem, as well as, its evolution. Collecting samples is one of the main ways to obtain information that can lead to effective measures against the phenomenon.

For most of these tasks, the robots will have to operate in a dynamic outdoors environment. The world around the robots changes and this can have an impact on the task execution. For example, in the underwater domain, a highly tidal

environment can develop strong currents that can affect the mission execution energy and time [4]. The robots must be able to cope with these changes and optimise their task execution online so that the mission efficiency is maximised.

A way to further improve the performance for such tasks is to introduce multiple robots solving the given problem. Having a robotic team performing a mission can have multiple benefits. One straightforward benefit is the parallelisation of tasks that can be achieved. For example, when a mission requires multiple points of interest to be inspected, each robot can inspect some of them. In the case of sampling, multiple robots can be used to collect samples from one or more areas. This can lead to faster mission execution times. An extra benefit is that the mission becomes robust to failures. Having one robot performing the mission can compromise the mission success in case the robot fails. With multiple robots, robot failures can be tolerated as other members of the team can still accomplish the tasks, with an expected penalty in time.

When robots collaborate in teams one issue that arises is the one of team coordination. To coordinate the robots must decide which tasks will be completed by which robot. This problem is referred to as the *Multi-Robot Task Allocation* (MRTA) problem [5]. One issue that is mentioned in [6] is that many of the MRTA approaches are heavily reliant on communication among the robot team. However, communication during field operations can be a scarce resource. For example, when operating teams of underwater robots, acoustic communications are the most popular choice. This type of communications is known to be lossy, low bandwidth and high latency [7]–[9].

This work focuses on missions where teams of robots have to achieve inspection or sampling tasks. In inspection tasks, the team has to visit various points of interest to collect data regarding a structure. In sampling tasks, the team has to collect samples from one or more regions in an area for estimating a phenomenon. These tasks can be present in any domain, be it land, air or sea. As an application scenario in the scope of this work, the underwater domain will be used. This choice is motivated by the fact that the underwater environment poses all the limitations that are already mentioned.

1.1 Research objectives

This work aims at expanding the field of multi-robot systems by introducing methods for optimising multi-robot mission execution in the presence of uncertainties and constraints in communication, as well as the available budget of the mission (i.e. available energy, time etc.). The goal is to provide solutions that can be applied online and that they can be robust to the aforementioned conditions. Specifically, this work addresses the following questions:

- How to address the multi-robot task allocation problem under communication constraints?
- How to optimise mission performance for sampling tasks under budget constraints in an online manner?

These questions are relevant to the limitations mentioned in the literature. Firstly, the MRTA problem can be heavily reliant on communication [6]. Secondly, the state of the art solution methods for sampling under budget constraints are computationally expensive and cannot be applied in an online manner [10], [11]. This work is an attempt to address both problems, as well as a combination of them that allows the solution of the multi-area sampling problem under budget constraints.

1.2 Methodology

This work focuses on providing solutions for the MRTA and online optimisation of mission performance under communication and budget constraints. Initially, the framework for exchanging semantic information over lossy and latent communication links presented in Maurelli and Saigol in 2012 [12] was studied. It was then extended with relevant semantic information to accommodate solutions to the MRTA problem. Subsequently, the MRTA problem was formulated as a *Mixed Integer Linear Problem* (MILP) and was solved using a commercial solver [13]. The setup was interfaced with actual land robots so that the MRTA could be tested in more realistic conditions. The communication latency and errors were simulated using a communications simulator developed by the Ocean Systems Laboratory.

Experiments were performed, and they validated the correctness of the MRTA approach.

The next phase of work addressed the problem of sampling mission optimisation under budget constraints. The first step was to implement and test in simulation the approach proposed in the work of Yu and Rus [10], [11]. The results showed that the method was not able to be applied online. The next step was to develop a heuristic method that would allow online application. This method was initially developed for a single vehicle. It was tested in simulation and found to perform near optimally, but requiring much less computational effort. It was then extended to allow multi-vehicle teams and its correctness was validated in simulation.

Finally, the two methods were merged to provide a novel method for long range, multi-vehicle sampling missions under budget constraints. In that problem, multiple areas had to be visited to be sampled. The problem was solved in two levels. The first level was solving an MRTA for assigning areas to vehicles, while the second level was run on each vehicle solving a single vehicle sampling optimisation for each assigned area. This method was compared with the multi-vehicle method that was previously developed in simulation. It was shown that it can perform better and required even less computational resources.

1.3 Contributions

In the scope of this work the following contributions were made:

1. A system for optimal multi-robot task allocation that works over a latent and lossy communications network.

2. A near optimal online heuristic for optimising sample collection missions under budget constraints that are performed by one or more vehicles.

3. A method for optimising multi-area sensing missions under budget constraints for teams of vehicles.

In general, the results collected through this work showed that the proposed methods allow running near optimal missions even under communication constraints. An overview of the proposed system architecture for a single vehicle can

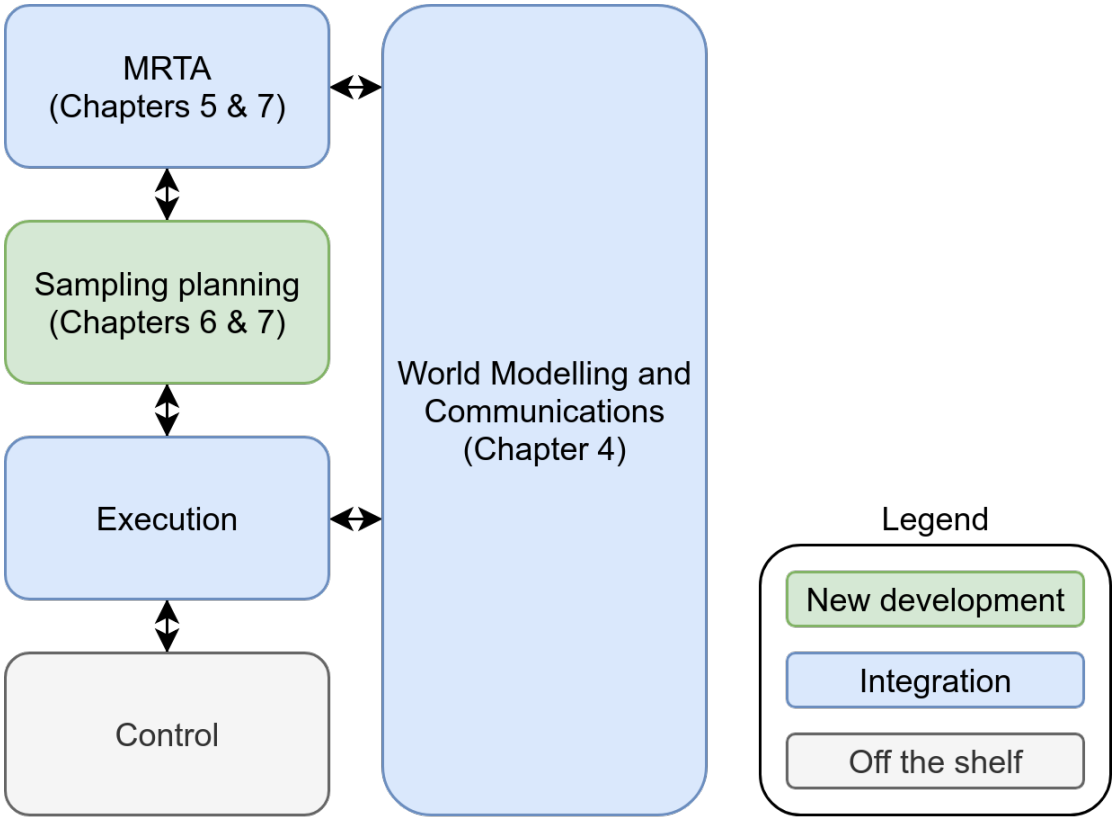


Figure 1.1. The system architecture for a single vehicle performing a multi-area sampling mission. Colour coding is used to denote newly developed parts or parts that were modified in the scope of this work.

be seen in figure 1.1. Colours are used to denote newly developed parts, as well as, parts required modifications in the scope of this work. In parentheses are the chapters describing or using the respective modules.

1.4 Structure

The rest of the work is organised as follows. Chapter 2 formally defines the studied problems. Chapter 3 presents a detailed literature review of the MRTA and sampling optimisation problems. In chapter 4 the system architecture is presented. Details are provided regarding the software architecture that allows communication among the team members and its integration with the real robots.

Chapter 5 presents the MILP formulation of the MRTA problem and the inspection mission that is used to evaluate it. Finally, it presents and discusses the results collected.

In chapter 6 the sensing mission optimisation is presented. It includes details

regarding the optimal solution methods for single and multiple vehicles operating under budget constraints. Moreover, the heuristic methods for one and multiple vehicles are presented in detail. Finally, results regarding the sensing mission optimisation under budget constraints are presented. A tuning method used to tune the heuristic parameters is provided, and the results regarding the single and multi-vehicle simulated experiments are discussed.

Chapter 7 presents the 2-level optimisation method for long-range sampling missions under budget constraints performed by multiple vehicles. Initially, it gives details on how the method is implemented. Then it describes the experimental setup emphasising on a method to generate random missions in order to evaluate the approach. It then provides the experimental results and discusses the findings.

Finally, chapter 8 summarises the contributions of this work. It discusses any limitations and provides future research directions that would allow the evolution of the methods to solve a wider area of problems.

Chapter 2

Problem definition

As briefly mentioned in the previous chapter, this thesis studies the optimisation of the behaviour and performance of teams of robots with an application to underwater vehicles. This chapter will present the theoretical background of the main problems studied by this work. Specifically, section 2.1 will present the *Multi-Robot Task Allocation* (MRTA) problem with an example, while section 2.2 will detail the sample collection problem.

2.1 The *Multi-Robot Task Allocation* (MRTA) problem

One of the fundamental questions faced by the designers of systems composed by more than one robot is *"Which robot should execute which task in order to cooperatively achieve the global goal?"* [5].

In multi-robot systems literature, there are two different approaches to achieve coordination. In the first approach, the coordination is achieved in an emergent way through the interaction of the team members with each other and the environment but without an explicit decision on task allocation. An example of this paradigm is the swarm robotics and their applications as seen in [14]. In the second approach, the team cooperates explicitly, deciding about the tasks that should be completed by each member, thus showing an intentional coordination behaviour [15].

In the scope of this work the latter approach is studied. In that approach a method to decide which tasks are executed by which robot is required. Specifically,

given a team of n robots $R = r_1, r_2, \ldots, r_n$, a set of m tasks $T = t_1, t_2, \ldots, t_m$ and a set $n \times m$ of robots' utilities $U = u_{1,1}, \ldots, u_{1,m}, u_{2,1} \ldots, u_{n,m}$, the *multi-robot task allocation* (MRTA) problem can be described as the optimal allocation A of the set of tasks T to a subset of robots $R' \subseteq R$ such as:

$$A : T \rightarrow R' \tag{2.1}$$

The following sections 2.1.1 and 2.1.2 will present in detail the concepts of tasks and utilities. In addition, a formal taxonomy of MRTA problems along with their mathematical formulations, will be presented in section 2.1.3. Finally, section 2.1.4 will provide an example of a task allocation problem.

2.1.1 Task

Given the existence of a global goal that has to be achieved by the robot team, a task can be defined as a sub-goal necessary for achieving the overall goal. Some of these goals can be represented by single actions performed by a single robot, while others can be decomposed into sub-tasks which can be allocated to different agents.

If a task t can be represented by a set of sub-tasks σ_t combined by a set of relationships ρ_t is a *decomposable* task, with (σ_t, ρ_t) being its *decomposition*. In cases where there is more than one decompositions for a task t, it is said to be *multiply decomposable*. According to [16] a task can be classified in one of the following categories:

Elemental task: *Elemental task* is a task that cannot be decomposed. It can also be referred to as *atomic task*.

Decomposable simple task: A task that can be decomposed in elemental tasks or decomposable simple subtasks is referred to as a *decomposable simple task*. A decomposable simple task or any of its decomposable simple subtasks can be only allocated to a single agent.

Simple task: A *simple task* is either an elemental task or a decomposable simple task.

Compound task: If a task can be decomposed into simple tasks or compound sub-tasks in a fixed unique way is called a *compound task*. Its decomposition can be allocated to multiple agents.

Complex task: If a task can be decomposed into a set of simple, compound or complex sub-tasks in multiple ways, is called a *complex task*. A complex task's decomposition can be allocated to multiple agents.

It should be noted that the optimal decomposition of a compound task can be determined in advance, while an optimal decomposition of a complex task can only be found at the time of the task allocation process. This fact makes the problem harder to solve as there are multiple ways to solve the same problem that need to be explored in order to find the optimal one.

Once a task is decomposed, dependencies and constraints arise among the sub-tasks. These need to be satisfied in order to have a feasible task allocation. For example, some tasks may be only performed by specific robots based on their capabilities. Other constraints may require tasks to be performed in a specific sequence or at a specific location. Such constraints may implicitly define the type of task that is to be executed.

2.1.2 Utility

Task allocation, being an optimisation problem, tries to find a feasible assignment of tasks to robots in a way that optimises an objective function. Such an objective function can be described by the utility of the robots performing the tasks. In [17] utility is described in terms of quality and cost and can be seen in (2.2).

$$U_{rt} = \begin{cases} Q_{rt} - C_{rt}, & \text{if r is capable of executing t} \\ -\infty, & \text{otherwise} \end{cases} \tag{2.2}$$

It can be seen that if a robot r can perform task t the utility U_{rt} is equal to the quality of the task performance Q_{rt} minus the cost for performing the task C_{rt}. If the robot cannot perform the task the utility is $-\infty$. This definition can be generalised for groups of robots and tasks. Suppose that $\mathcal{R}$ represents a subset of the robots in the team with $|\mathcal{R}| \geq 1$, and likewise $\mathcal{T}$ represents a subset of tasks with $|\mathcal{T}| \geq 1$. The utility measure for a sub-team of robots and a given subset of

task is defined as follows:

$$U_{\mathcal{RT}} = \begin{cases} Q_{\mathcal{RT}} - C_{\mathcal{RT}}, & \text{if } \mathcal{R} \text{ is capable of executing } \mathcal{T} \\ -\infty, & \text{otherwise} \end{cases} \tag{2.3}$$

In a similar way to (2.2), when a sub-team of robots $\mathcal{R}$ can perform a subset of tasks $\mathcal{T}$ the utility is the difference between the quality $Q_{\mathcal{RT}}$ and cost $C_{\mathcal{RT}}$, if not the utility is still $-\infty$. Building on that, one can define the effective utility, ${}^{e}U_{rt}^{\mathcal{RT}}$ for a robot $r \in \mathcal{R}$ and a task $t \in \mathcal{T}$ such as:

$$U_{\mathcal{RT}} = \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} {}^{e}U_{rt}^{\mathcal{RT}} \tag{2.4}$$

If the utilities in the given problem are *independent*, i.e. ${}^{e}U_{rt}^{\mathcal{RT}} = U_{rt}$, equation (2.4) becomes:

$$U_{\mathcal{RT}} = \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} U_{rt} \tag{2.5}$$

On the other hand, if the utilities of a given problem are *interrelated*, i.e. ${}^{e}U_{rt}^{\mathcal{RT}} \neq U_{rt}$, equation (2.4) becomes:

$$U_{\mathcal{RT}} \neq \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} U_{rt} \tag{2.6}$$

An example for this a case, is when the cost of performing two tasks changes based on the relative order of the task. The tasks can then be ordered in a way that minimises the cost and creates a synergistic relationship between them. The equation (2.6) then becomes:

$$U_{\mathcal{RT}} > \sum_{r \in \mathcal{R}} \sum_{t \in \mathcal{T}} U_{rt} \tag{2.7}$$

In other words, the effective utility ${}^{e}U_{rt}^{\mathcal{RT}}$ of performing task t in the context of the task subset $\mathcal{T}$ is greater than performing just the task t by itself.

2.1.3 Taxonomy

In addition to the task and utility definition, the work of [17] presents a taxonomy for classifying various MRTA problems based on the level of interdependence of agent-task utilities in a problem. It is claimed that the degree of interdependence shows how coupled a problem is and is a strong indicator of the difficulty to solve such a problem. Following are the four classes composing the proposed taxonomy:

No Dependencies (ND): In the first category there are task allocation problems involving simple or compound tasks which have independent agent-task utilities. According to (2.5), the effective utility of an agent for a specific task does not depend on any other task or agent. In such problems, the task decomposition is not related to the task allocation problem. Additionally, the order of performing the tasks does not affect the utility for an agent; thus there is no scheduling optimisation.

In-schedule Dependencies (ID): This category contains all these problems of simple or compound tasks, where the agent-task utilities have intra-schedule dependencies. In such problems, the effective utility of a task depends on which other tasks the agent is performing. As in the ND case, the task decomposition is independent of the allocation. On the other hand, there are scheduling optimisation problems, but each agent's scheduling problem is decoupled from the other agents' problems.

Cross-schedule Dependencies (XD): This category extends the ID by allowing agent-task utilities to have inter-schedule dependencies. This means that the effective utility of an agent for a specific task depends on the schedule of other agents, in addition to the agent's schedule itself. Again, the task decomposition is decoupled from the task allocation by allowing only simple or compound tasks, where the decomposition is optimally determined before the task allocation. Additionally, in this category, constraints are allowed between the schedules of agents; thus the schedule optimisation problems cannot be decoupled.

Complex Dependencies (CD): The last category includes all the problems in which the agent-task utilities have inter-schedule dependencies for complex tasks on top of potential in-schedule and cross-schedule dependencies for

simple or compound tasks. In this particular case, the agent-task utility is depending on the schedules of other agents in a specific way dictated by the current task decomposition. As already mentioned for complex tasks, the task decomposition must be optimised in parallel with the task allocation. In addition, there can be constraints between the schedules of different agents, so the schedule optimisation problem of one agent depends on the schedule optimisation of the other agents.

In addition to the above taxonomy, a second level can be added that can further classify the problems in more categories using the taxonomy proposed in [5]. In that work, the classification is based on task execution properties of a robot, task properties and task allocation properties. Specifically, a robot can be single or multi-task depending on its ability to execute tasks in parallel. Moreover, a task can be single robot if its successful execution requires one robot, while a multi-robot task requires multiple robots to be successfully executed. Finally, instantaneous assignment considers only one allocation based on the available information, while a time-extended assignment can plan for future allocations.

2.1.4 Example

In order to help the reader understand the properties of each taxonomy as well as its complexity, an example is presented. In this example, a problem is presented as it evolves from one taxonomy to another with increasing complexity.

Suppose that there are two robots R_1 and R_2 that are tasked to collect some resources from different locations, namely T_1, T_2, T_3 and T_4. Both robots start from a single location, called the depot and is where resources should be stored. In addition, each one can carry one resource. The example setting can be seen in figure 2.1. Each resource location and the depot are vertices. Weights on the edges connecting the depot vertex with the resources denote the cost to travel to the respective resource location.

In this setting, each robot has to travel to a resource, load it and move back to the depot to unload it. It can be seen that the utility of executing a task is independent of any other task. Such kind of problems fall into the ND category. A potential allocation can be seen in figure 2.2.

Assuming the same quality for each task the utility is merely relying on the cost.

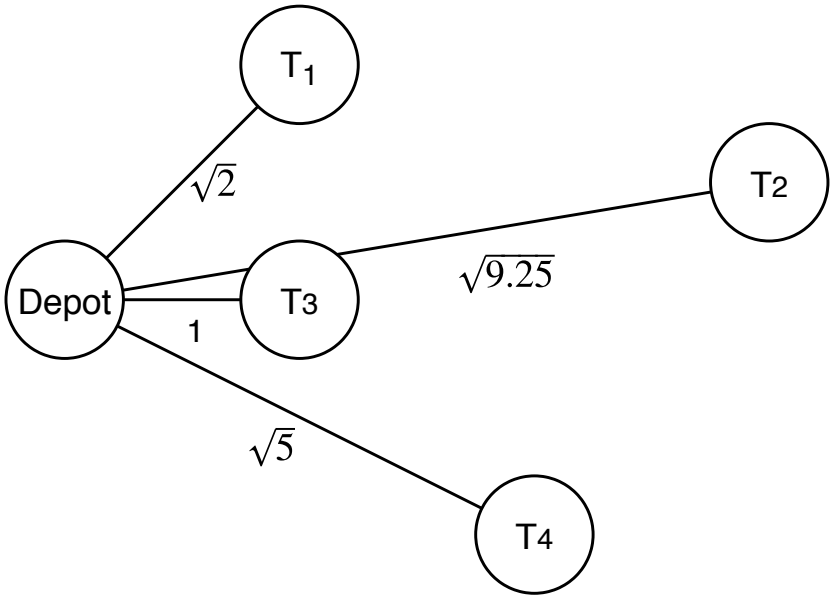


Figure 2.1. Example setting for a resource collection problem. Robots are tasked to collect resources from four different locations $T_1 - T_4$ and gather them at a depot location. The robots start and end their routes ah the depot. Weights on edges connecting the depot with each resource denote the cost to travel to the respective resource location.

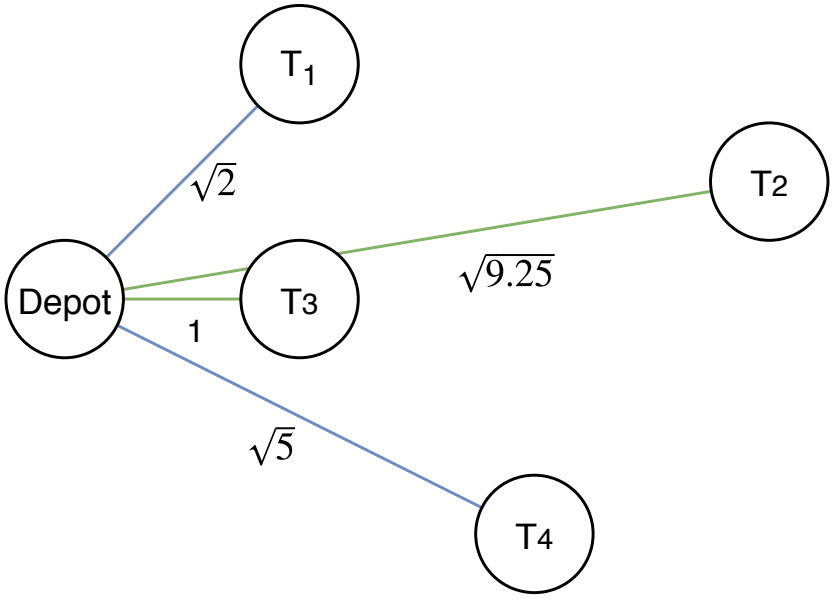
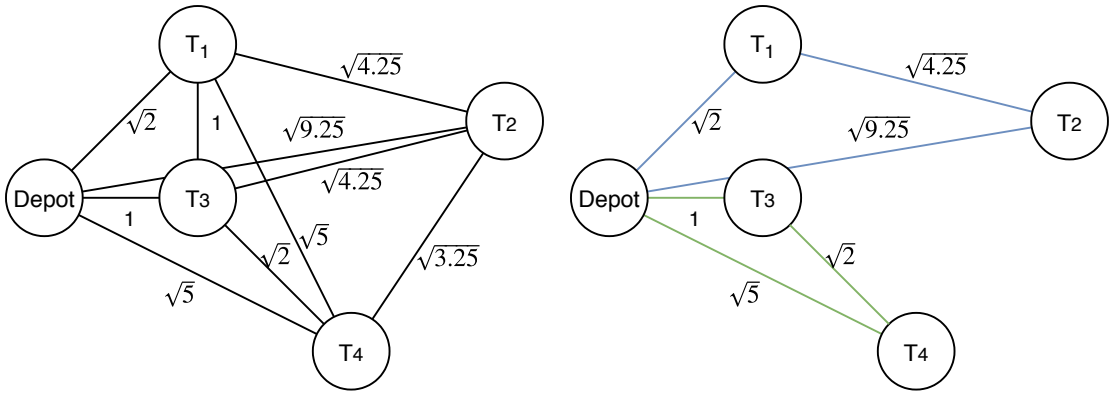


Figure 2.2. Example allocation for the ND case where each robot can only carry one resource. R_1 must fetch resources from T_2 and T_3 travelling 8.082 units, while R_2 has to travel 7.3 units from T_1 and T_4.



(a) Fully connected graph of tasks **(b)** MRTA solution

Figure 2.3. Example allocation for the ID case where each robot can carry two resources. Figure 2.3a shows the fully connected graph representing the problem, while figure 2.3b shows an optimal task allocation.

The above allocation tries to maximise the utility of each agent by minimising the total travelled cost. The first robot will have to visit T_2 and T_3, while the second one will have to visit T_1 and T_4. In this solution, the robots travelled 8.082 and 7.3 units respectively.

Wanting to reduce the resource collection effort, it is decided to upgrade the robots with more storage so that they can carry two resources at a time. Instantly, the task allocation problem becomes more complex as the effective utility of an agent for a task depends on other tasks this agent is executing and their relative order. This creates a problem of the ID category and a potential allocation can be seen in figure 2.3.

In figure 2.3a one can see the fully connected undirected graph that represents the problem. Weights on the edges represent the distance that needs to be travelled. Figure 2.3a shows an example allocation. As it can be seen the extra storage allows the robots to reduce the distance travelled and thus increase the gained utility. Specifically, R_1 will have to travel to T_3 and T_4 at a cost of 4.650 units while R_2 has to get the resources from T_1 and T_2 at a cost of 6.516 units.

In case some of the resources were of bad quality one would like to gather information for each one before trying to collect them. This can be done by adding a sensor to one of the robots, removing its ability to carry any resources. In this case, the problem is transformed into an XD category problem. Since one robot has first to sense the resource for its quality, the other robot has to wait

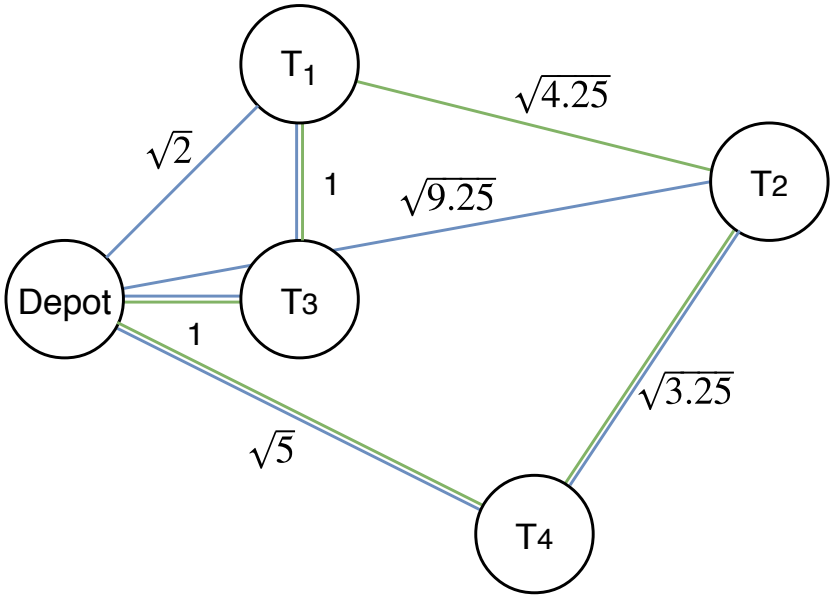


Figure 2.4. Example allocation for the XD case. The first allocation (green) visits all locations to collect data for each available resource. Then the second robot collects resources in two phases denoted by the blue line.

before collecting it. Also, the quality of the task performed by the collecting robot is influenced by the schedule of the other agent. An example solution can be seen in figure 2.4.

As can be seen, there are two distinct allocations. The first one for robot R_1, denoted by the green line, visits the resource locations to collect data. The second one for robot R_2, shown by the blue line, visits resource locations and collects resources as soon as they are evaluated by R_1. It can be seen that the collection paths are different from the one found for the ID problem in figure 2.3b, as they depend on the sensing agent's schedule.

Finally, to reduce the travelling of the sensing robot and the waiting time of the fetching robot, it is decided to add a ranged quality sensor to the sensing robot. This sensor has a limited range so that some of the resources will still have to be visited. The sensor addition transforms the problem to a CD problem as there are multiple decompositions for the same task. An example allocation can be seen in figure 2.5.

The first robot has to physically visit T_3 and T_2 to collect data regarding resource quality. While on T_3 it can collect ranged measurements from T_1 and T_4. Robot R_2 can proceed collecting the resources faster as it does not have to wait for R_1 to go to every resource.

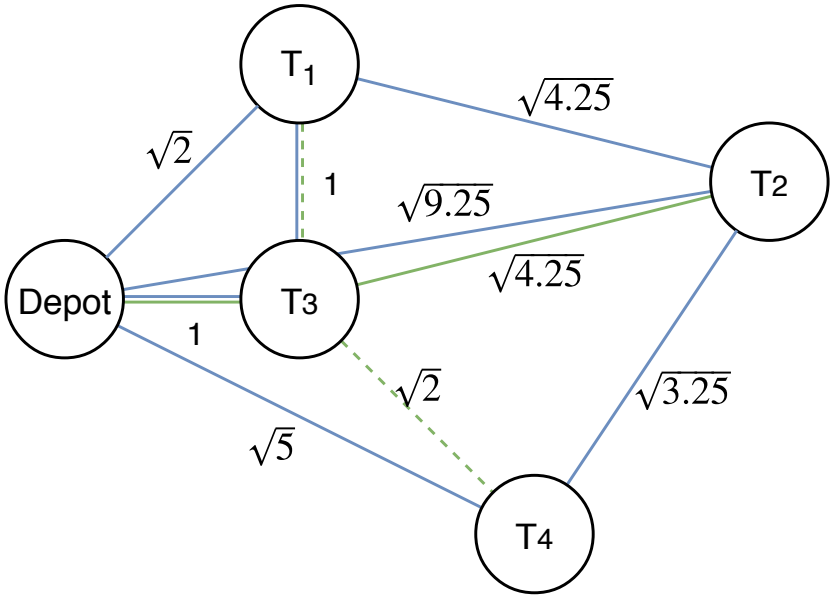


Figure 2.5. Example allocation for the CD case. In this allocation a decomposition where the sensing robot physically visits T_3 and T_2, while collecting ranged measurements from T_1 and T_4 is chosen.

2.2 The sample collection problem

The second problem studied by this work is the one of sample collection. In this problem, a robot, or a team of robots, has to collect samples from a predefined area so that a physical phenomenon can be studied. For example, phenomena like algal blooms, oil spills and chemical pollution need to be monitored and their extent to be monitored. All these types of sensing missions can be modelled as a scalar field that needs to be estimated [18].

A method to model and estimate such a field is presented in [19]. It is proposed to model the field in a probabilistic way using a *Gaussian process* (GP) based approach. To achieve that a grid of sampling points is used to approximate the field. The field values then can be modelled as a multivariate Gaussian distribution, where one dimension is used for each sampling point. The covariance of the distribution is calculated using a kernel function. One popular choice for a kernel is the square exponential kernel shown in (2.8).

$$k(x_i, x_j) = exp(-\frac{\|x_i - x_j\|}{2l^2}) \tag{2.8}$$

The kernel is used to model the drop in the correlation of the value of two points based on their distance. The parameter l governs that behaviour and can be estimated from previously gathered data. In general, the distance that two points can be correlated is longer, when l grows larger.

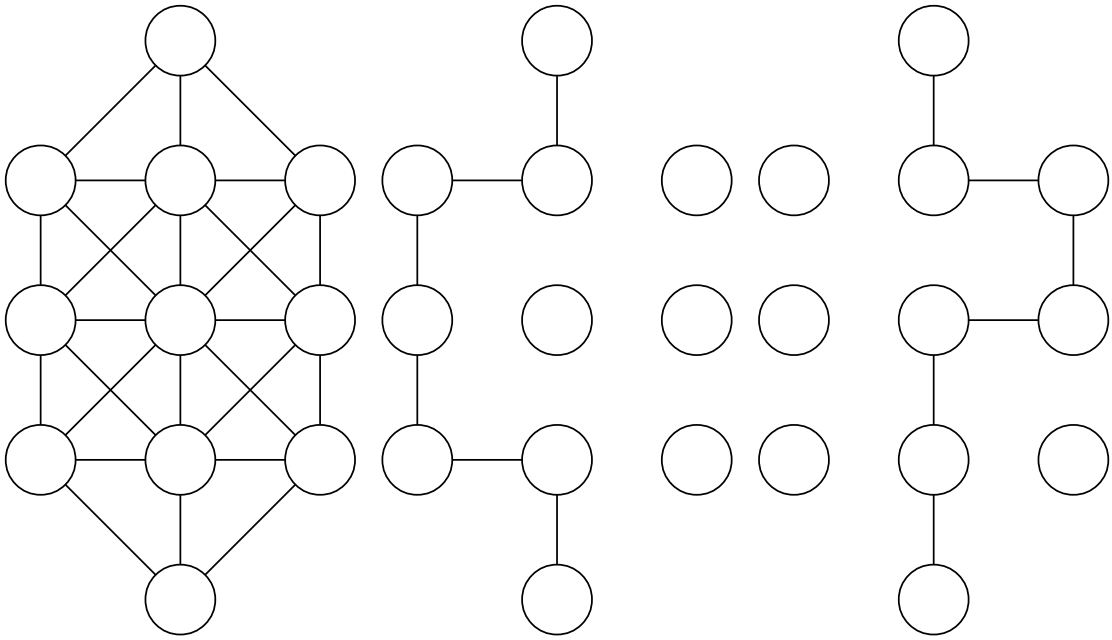
Given the above formulation, one can observe that the accuracy of the field estimation is dependent on the number of samples that are gathered. With more sampling points a better estimation can be achieved. Unfortunately, operating in the real environment, robots have to cope with its dynamic nature and many constraints it may impose on the mission execution. In addition, robots may have limitations in the sampling capacity, meaning that only a small number of samples may be collected. Given the limitations, it can be seen that the problem can be formulated as an optimisation problem that tries to maximise the information gathered by the sampling process.

In detail, let $S = s_1, s_2, \ldots, s_n$ be a set of sampling points over an area of interest, V is a set of vehicles that can collect samples having $|V| \geq 1$, and $R = r_1, r_2, \ldots, r_n$ a set of rewards for collecting samples. The optimisation problem is finding a subset $S' \subseteq S$ of samples that maximise the reward accumulations while being able to be visited by the vehicles V given any constraints.

Revisiting the utility theory presented in the previous sections one can represent the utility of collecting one sample u_i to be equal with the reward r_i minus the cost of collecting that sample c_i. As a cost one can use the distance that needs to be travelled to reach the sampling point. Given the spatial distribution of the sampling points, the order of visit affects the cost of sampling. It can be seen that the utilities are interrelated forming a scheduling optimisation problem.

2.2.1 Example

An example for the sampling problem can be seen in figure 2.6. The area to be sampled consists of 9 vertices being in the centre of figure 2.6a. The vehicle has to start its sampling mission from the lowest vertex and finish at the topmost vertex. The edges represent the distance required to travel between vertices.



(a) Fully connected graph for the sampling problem.

(b) A candidate solution for the sampling problem.

(c) Another candidate solution for the problem.

Figure 2.6. A sampling problem of a 3-by-3 area along with two solutions. The robot has to start from the bottom vertex and finish at the top vertex. It must visit as many vertices as possible without violating the budget constraints.

In this example, the resource that is constrained is the allowed travelled distance. If the maximum allowed budget is set to 6 units solutions such as the ones shown in figures 2.6b and 2.6c can be found. It should be noted that these solutions are equivalent, as the amount of visited vertices is the same.

2.3 Summary

This chapter presented the theoretical background for the problems studied in the scope of this work. Specifically, it presented the *Multi-Robot Task Allocation* (MRTA) problem, along with a taxonomy from the literature based on the properties of the tasks to be allocated. A comprehensive example of an MRTA problem evolving from one taxonomy class to another was detailed.

Moreover, the sample collection problem was presented. A formulation for estimating a scalar filed based on Gaussian processes was detailed, and the constraints of a sampling mission were presented. Finally, an example of a sampling

mission, along with potential solutions was given. The next chapter will provide a detailed literature review on both problems.

Chapter 3

Literature review

The previous chapter presented the theoretical background regarding the MRTA and sample collection problems. This chapter will present relevant solution methods from the literature. Section 3.1 will present approaches to address all different classes of MRTA problems, while section 3.2 will detail the previous work regarding the sampling problem.

3.1 MRTA problem literature

As mentioned in the previous chapter an MRTA problem is classified into four distinctive classes based on the level of interdependence of the task utilities and the task decomposition. Over the years, mathematical models, as well as solution methods for the problems have been presented in the literature. This section aims at thoroughly presenting them in the following sections.

3.1.1 No Dependencies

The first category represents tasks whose effective utility for an agent is not depending on any other agent or task. According to the work of [5], a mathematical formulation for this category of problems can be found in the combinatorial optimisation literature and specifically to the linear assignment problem presented in [20]. Linear assignment problems can be solved in polynomial time using the Hungarian algorithm presented in [21]. One of the first MRTA approaches addressing the ND problem is presented in [22]. In this work, the multi-robot exploration and mapping problem is addressed. Task allocation is performed using a cen-

tralised auction scheme. A distributed auction system for MRTA is presented in [23]. The presented system, called *MURDOCH*, uses auctions to coordinate heterogeneous teams of robots performing various tasks. It is tested in both loosely coupled and highly coupled tasks. In the first category, tasks can be performed by a single robot, such as object-tracking or sentry duty. In the second category, a box-pushing task is used, that required two robots to push a box. A different method is used in the work of [24], where potential fields are used. There, the role assignment problem in a robot football scenario is studied. In this approach, the assignment problem is continuously solved at a given frequency. This allows robots to change roles dynamically based on the evolution of the game.

3.1.2 In-schedule Dependencies

The ID category includes tasks that their effective utility is influenced by other tasks in an agents schedule. Formulations for these kinds of problems can also be found in the combinatorial optimisation literature. A first formulation is presented in [25], [26] which describe the generalised assignment problem. Another formulation is addressing the machine scheduling problem as presented in [27]. Formulations addressing various flavours of the *Vehicle Routing Problem* (VRP) appear in [28], while the *multi-Traveling Salesman Problem* m-TSP is studied in [29]. A common characteristic of all these problems is that they are all NP-hard. That makes them much harder to solve than the problems in the ND category.

The ID category has been extensively studied in the MRTA literature. One of the first approaches presented the GRAMMPS mission planner in [30]. In this work, the mission is defined as TSP and m-TSP components, and the task allocation is performed using a mixture of exhaustive and randomised search. Another work that uses a mixed-integer formulation to address the MRTA is presented in [31]. The presented formulation is close to the one for the m-TSP. To solve the problem, an auction-based method is used that performs repeated single-item auctions that allocate targets to agents.

Auction-based task allocation methods are extensively used to solve MRTA problems. They are well suited to robot teams as the allocation procedure can be performed in a distributed way. One of the pioneering approaches is the work of Diaz [32], where an auction-based approach, named *TraderBots*, is used for

coordinating a multi-robot team. In this approach, agents use auctions for task allocations by placing bids on tasks. Each agent has only local information regarding his schedule of tasks that is committed to perform. To provide good solutions, the system allows periodical reauctioning of already allocated tasks. This helps to escape local minima by allowing better schedules to be planned for each agent. The application presented requires a team of robots to travel to specific locations and collect sensor information. This problem can be formulated by the multi-depot m-TSP. In the presented case agents formulate bids by using the cost increase occurred by inserting the new task into their schedule plus a percentage of their profit expectations for executing the task. In general, the presented results provided reasonable solutions, but without any guarantees on optimality.

Another auction-based method is presented in [33]. The problem solved was an exploration task where robots have to visit some predefined points in the environment. Task allocation is performed using combinatorial auctions. In such auctions, bundles of tasks are offered instead of single tasks. The agents then bid on these bundles given their total profit minus the total cost.

A combination of the previous two methods is presented in [34]. That work aims at reducing the complexity of determining the winner for an auction compared to [33]. It is achieved through sequential bundle-bid single-sale auctions. In these auctions, agents select and bid in bundles of various sizes, up to a specified size k. The robot acting as an auctioneer is then allocating k additional tasks to the agents. This process is repeated until all tasks are allocated.

A theoretical analysis of auction-based methods for multi-robot routing is provided in [35]. The performance of the auction-based methods is evaluated using three different objective functions. The first tries to minimise the sum of the cost of the robot paths (MINISUM), the second minimises the maximum path cost (MINIMAX), while finally, the third minimises the average path cost (MINIAVE). In addition to the objective functions, appropriate bidding rules are provided for each one of them. Finally, approximate bounds on the performance of the auction based methods are provided.

3.1.3 Cross-schedule Dependencies

The XD category addresses problems in which the effective utility of an agent performing a task depends on the schedule of that agent, as well as on the schedules of other agents in the system. This behaviour can be observed on two different occasions. In the first occasion, there are single agent tasks that can be allocated to different agents but are related by some constraint. Example of constraints can be precedence, i.e. when one task must be completed before the other can start, and proximity, i.e. when two tasks must be performed at a specific distance from each other. In the second occasion, tasks require more than one agent in order to be completed. Each of those tasks needs to be allocated to a specific subset of the available agents. Such a problem is referred to in the literature as the coalition formation problem. It can be seen that in the XD category the agents cannot optimise their schedules without coordinating with the rest of the team. On the contrary, the ID category allows each agent to optimise its schedule independently.

Problems with single-agent tasks having cross-schedule dependencies can be modelled using a generalisation of the assignment problem by introducing additional constraints as proposed in [36]. For the specific case where precedence constraints exist, the combinatorial optimisation literature proposes models for solving machine scheduling problems [27] or vehicle routing problem [37]–[39]. VRPs are of particular interest to the field of robotics as these models incorporate travelling times and costs that are directly applicable to mobile robotic applications.

In the MRTA literature, there are some approaches that address the single-agent task with cross-schedule dependencies problem. An initial approach is presented in [40], where the M+ system addresses the task allocation problem using a market-based approach based on iterated instantaneous assignment. Precedence constraints are satisfied by permitting negotiations only on tasks that have all their preconditions met. Another market-based approach is presented in [41]. In that approach, multiple tasks that have constraints between them are auctioned simultaneously. The bidding agents submit bids that are expressed as functions of constrained variables. A cost minimisation algorithm is used to perform the task allocation. This algorithm decides which agent is going to perform the task, as well as the value of the constrained variables. This method satisfies precedence con-

straints between tasks but cannot perform time extended allocation. This means that each agent is awarded only one task that needs to be executed and a schedule cannot be formed for each agent.

The work of Chien et al. [42] presents three different approaches for addressing the MRTA in a distributed geological science scenario. In this scenario, there are resource constraints for each robot, as well as constraints imposed by shared resources. All three approaches are based on the ASPEN planner. In the first approach uses the centralised ASPEN planner to compute a schedule for the whole team. The planner uses various heuristics to create the schedule; for example, it uses heuristics from the m-TSP literature combined with iterative plan repair algorithms. The second approach divides the shared resources equally among the members of the team and allocates goals to them. Each member then uses the ASPEN planner to create a schedule in a decentralised way. The final approach uses an auction-based method where each of the robots uses the ASPEN planner to calculate its bid for each task.

The method proposed in [43] allows time-based ordering of pairs tasks between two robots. If first auctions a task to a robot, which is designated as the "master". The "master" then determines the start time for that task. This time is then used to determine the start time of the dependent task which is auctioned to a "slave" robot. This "master"-"slave" relationship is preserved throughout the execution of both tasks. The pair maintains communication to transmit necessary information regarding dynamic changes to the environment that may affect the plan. Such changes can cause the tasks to be rescheduled or reauctioned to other robots.

The problem of coordinating a team of robots while maintaining connectivity is studied in the work of [44]. In such a scenario the robots have limited communication range. This communication constraint is creating dependencies among the schedules of all the robots. Several task allocation methods are proposed, including a greedy and an auction-based one. Connectivity among robots is maintained using a specific navigation mechanism.

The second type of cross-schedule dependencies involves tasks that require multiple robots for their successful execution. Finding which subset of robots should perform a task can be modelled as a coalition formation problem. In the combinatorial optimisation literature, the coalition formation problem can be modelled as the set-partitioning problem or as the set-covering problem [45]. In

the first case, the robots can execute only one task at a time and therefore be part of a single coalition at any given time. The second case refers to robots that can execute more than one tasks simultaneously and can be part of multiple coalitions at the same time.

In some cases, multiple multi-robot tasks exist that require scheduling. This problem is addressed in the work of [46], where a mixed-integer programming formulation is presented for the *Coalition Formation with Spatial and Temporal Constraints Problem* (CFSTP). The authors also present anytime heuristics for a fast solution to the problem.

The coalition formation problem has been studied extensively in the MRTA literature. A particular example can be found in the work of Shehory and Kraus [47] where an approach to address the MRTA for single and multi-agent tasks is presented. The presented scenario requires goods of different sizes and weights to be transported by a team of agents. Some of the goods are small enough to allow a single agent to transport them, while the size or weight of others requires more than one agent. For the successful execution of all the tasks agents are required to form coalitions. To solve this problem, it is proposed to use a greedy, distributed, anytime set-partitioning algorithm. Each formed coalition has a set of available capabilities that should match the set of required capabilities of a given task. The utility of the coalition performing a task is the sum of the individual utilities of the agents performing the task. The proposed algorithm has two stages. In the first stage, the agents compute the coalition values in a distributed way. In the second stage, coalitions are matched with tasks iteratively. An extension of this work is presented in [48] where a distributed set-covering algorithm is used to solve the problem where agents can be part of more than one coalitions. In addition, the proposed approach allows tasks with precedence constraints by computing coalitions for all the prerequisite tasks of a given task.

The approach of [47] and [48] is designed with abstract agents in mind. This is reflected in the design requiring lots of communication and not considering which capabilities should appear in the configuration of a single agent when considering it for a coalition. Such properties make the adoption of that approach in the robotics domain hard. The work presented in [49] addresses the aforementioned issues by adapting the approach of Shehory and Kraus. Moreover, it introduces additional constraints to discourage the formation of imbalanced coalitions.

An auction-based approach to the coalition formation problem is presented in [50]. In this approach when a robot discovers a task becomes the task owner and creates an auction trying to recruit other robots to form a coalition. Another auction-based approach is presented in [51] where combinatorial bids are used for coalition formation. The approach uses capability vectors to model task requirements and agent capabilities. Each task has a "manager" that creates an auction. Each of the other agents bid on the task using their capability vectors. The managing robot selects a subset of robots that satisfy the capability requirements of the task and informs them with a "task pre-award message". The subset of agents then coordinates and creates a "bidding combination" that is transmitted to the task manager. The task is then awarded to the formed coalition.

A more comprehensive approach to the multi-task robots is presented in [52]. In the described problem there are single and multi-robot tasks, and the robots can perform multiple tasks in parallel limited by their local resources. Examples of resources are the robot's communication link, the processor capacity and the robot's position. A robot can complete a task using its local resources or resources of other robots. Coalition formation happens by requesting data or resources to complete a specific task. To solve the problem, single-round auctions are used. Each auction has two stages. The first stage forms potential coalitions by requesting resources from the team. The second stage determines the best coalition and awards the task.

The time-extended coalition formation for single-task robots is studied in [53]. The coordination problem is modelled as a constraint optimisation problem formulated as a *mixed-integer linear program* (MILP). The proposed solution, named *Constraint Optimisation Coordination Architecture* (COCoA), has a heuristic method that produces an initial solution. This solution is then given to a commercial linear programming solver for further optimisation.

Another work using MILP formulations to address the time-extended coalition formation is presented in [54]. This work also focuses on single-task agents and a task may require multiple agents to be able to be completed. The problem can have various temporal and location constraints. Examples are task precedence, synchronisation, task proximity and location capacity constraints. To model the problem a set-partitioning MILP formulation is used. A proposed planner, named *xTeam*, provides the solution to the problem. This planner utilises a custom

branch-and-price algorithm [55] to solve the given coordination problem.

3.1.4 Complex Dependencies

The final category of problems is involving complex tasks. Such tasks have multiple potential decompositions, of which at least one can be allocated to multiple agents. In such a problem one must decide not only which robots should perform each task and when, but also which decomposition should be used. In the MRTA literature, only a few approaches address the CD class.

The work presented in [56] addresses time-extended MRTA problems with intra-path constraints. An example application is a disaster-response scenario assigning fire sources to fire-fighting robots. The robots can reach the fire sources using more than one routes. Some of these routes are blocked by debris which can be cleared using bulldozer-type robots. To calculate the cost of a route depends on whether it will be clean of debris. This creates complex dependencies as multiple allocations must be decided simultaneously. Specifically, it must be decided, which fire-fighting robots will be allocated to which fire, which routes they will take and which bulldozer-type robots will be assigned to clear which debris along those routes. To solve this problem, the authors propose two different methods. The first method uses an auction-based approach along with task clustering and opportunistic path planning. This approach performs a search in the space of possible time-extended schedules and allocations. The second method is based on a genetic algorithm trying to find a valid plan.

The work presented in [57] proposes a solution method for complex problems with tasks requiring multiple robots. This method is based on schemas which are structures having input and output ports, local variables and a behaviour. To construct a solution, this method is dynamically connecting a network of schemas that are on the robots of the team. The connection of various schemas given their input and output ports can produce a behaviour that solves the given problem. It can be seen that connecting different schemas in different ways can produce multiple decompositions for solving a specific task. The proposed solution method is called *ASyMTRe* and is a greedy method that performs a search over the different schema configurations trying to find a solution. The authors also provide a distributed version called *ASyMTRe-D* where each robot decides what information

is needed and requests it from other members of the team.

An auction-based approach for solving time-extended task allocation problems of complex tasks is presented in [16]. This work extends the work presented in [32] by trading task trees instead of simple tasks. Each task tree represents a potential decomposition for a given task. During an auction, robots are allowed to change the decomposition of the task they are biding for. It is also allowed to bid only for parts of a tree, thus allowing the formation of coalitions. After all the robots place a bid, the auctioneer selects those bids that satisfy the task requirements and have the lowest cost for the whole team. These robots are then awarded the task.

3.2 Sampling problem literature

As described in the previous chapter the sampling problem can be formulated as an optimisation problem. In this problem, one wants to maximise the utility gathered by an agent, or a team of agents, while respecting some resource constraint. The following sections will present formulations from the combinatorial optimisation literature as well as heuristic solutions for the problem.

3.2.1 The orienteering problem

In the combinatorial optimisation literature, there are various approaches to address this problem. The first work to address this problem is presented in [58]. In this work, a heuristic optimisation method is applied to the game of orienteering. In this game, there are various control points, each one having a score, and a limited amount of time. Each contestant has to maximise its score by visiting the control points given the time limitation. The term *Orienteering Problem* (OP) widely used in the literature was first coined in the work of [59]. This work provides a heuristic solution method that improves the results of [58]. The first work presenting an exact method for solving the OP is shown in [60]. There, an ILP formulation is presented, along with upper and lower bounds and an exact enumerative algorithm. Another exact solution is presented in [61]. This work expands the work of [60] by adding valid inequalities to the problem definition. The optimal solution is found using a branch-and-cut method. Applications of

the problem to transportation and logistics systems are presented in the review papers of [62], [63].

Since the OP is an NP-hard problem, an exact solution can be difficult to find. For that multiple heuristic methods have been proposed in the literature. In [64] a five-step heuristic is presented. This heuristic initially constructs several paths using greedy initialisation. Then it iteratively performs operations of two-point exchange between paths, one-point move from path to path, 2-opt optimisation and reinitialisation of the best-found path by removing the lowest-performing control points, or vertices, of the path. Their approach performed better than the state of the art in multiple cases.

The work presented in [65] uses a *Tabu Search* (TS) heuristic to find near-optimal solutions for the *Selective Travelling Salesman Problem* (STSP). The STSP is a version of the OP where a maximal profit Hamiltonian cycle is searched over a subset of the vertices. In general, the OP searches of a Hamiltonian path since starting and ending positions may be different. Their proposed heuristic manages to provide results that are close to optimal for the used test cases.

The work of *Liang et al.* [66] presents two heuristics for the OP. The first heuristic is based on *Ant-Colony optimisation* (ACO) where virtual ants construct paths using trails of pheromone. The second presented heuristic is based on TS. The proposed methods are compared against a genetic algorithm approach [67] and the heuristic of [64]. The performance of both presented methods is on-par with the state of the art in solution quality. In terms of computational needs, they were both much faster than the genetic algorithm. Compared to the heuristic of [64] the TS heuristic performed comparably, while the ACO algorithm was two times slower on average. Nevertheless, the ACO method proved to be more stable having a lower standard deviation in the results than the TS heuristic.

Another work presenting two different heuristic methods is presented in [68]. In this case, it is used to provide solutions to a multi-objective orienteering problem. The application presented is one of planning individual tourist routes in a city based on the preferences of the tourist. For example, some tourists prefer cultural points, while others have a higher preference for shopping. Therefore a multi-objective approach is required. The first presented method is using ACO optimisation, while the second is using a *Variable Neighbourhood Search* (VNS) approach. The proposed approaches are tested on benchmark instances and in

real-world applications. Regarding their single-objective performance, they are compared with approaches from the literature and perform equally or better. For the multi-objective case, the ACO based approach usually performs better than the VNS approach.

One of the first *Genetic Algorithm* (GA) approaches for the OP is presented in [67]. In this approach, an adaptive penalty function is used that allows considering infeasible solutions as solution candidates. It is compared against the results of [64] and an artificial neural network optimiser. It manages to produce better results than the state of the art in four instances.

In the work presented in [69] a GA is used to solve the *Generalised Orienteering Problem* (GOP). In the GOP each of the vertices has a score based on multiple attributes, and the objective function of the problem is non-linear. It uses an elitist GA that solutions do not degrade during the optimisation process. The application is again one for tourist planning. Specifically, tours are found through 27 cities in China, where each city's score is based on four metrics, namely natural beauty, historical significance, cultural and educational attractions, and business opportunities. The proposed approach is compared to an artificial neural network specifically designed to tackle this problem. Both approaches perform comparably in terms of tour quality, but the GA is much faster. To solve

To solve the OP in large instances, a GA is proposed in the work of *Karbowska et al.* [70]. Their approach evaluates each solution based not only on the collected rewards for visiting vertices but also based on the total cost of the solution. The presented approach is compared against a *Guided Local Search* (GLS) method and another GA approach. It manages to outperform the other methods both in terms of profit and in terms of computational time.

A method that found improved or, at least, equally good solutions to the state of the art is presented in [71]. The proposed approach is based on a discrete strengthened particle swarm optimisation method. This method provides higher robustness as the quality of the found solutions shows low variance. It is compared against multiple different heuristic approaches from the literature and finds at least same quality paths as the state of the art. In terms of computational complexity, it is on lower than methods providing equal solution quality.

The work presented in [72] uses a *Greedy Randomised Adaptive Search Procedure* (GRASP) with *Path Relinking* (PR). The authors present four different

methods based on GRASP to generate a population of solutions. These solutions are then improved using PR between them. The proposed approach is compared against heuristics in standard problems from the literature and shows comparable performance in terms of score requiring a fraction of the computation time. Another GRASP based method for the OP appears in [73]. There the GRASP method is used to construct a population of solutions that are then improved using local search methods and an evolutionary algorithm. The proposed approach manages to find the solutions that match the best solutions found in the literature.

3.2.2 The team orienteering problem

An extended version of the OP for multiple agents is presented in [74]. The problem is named *Team Orienteering Problem* (TOP) and consists of multiple agents starting from a single place trying to collect the maximum reward using a limited budget. Only one agent may visit each vertex providing a reward. It can be seen that the standard OP can be considered a special case of the TOP. The work presented in [74] provides a heuristic approach for the TOP that is similar to the one presented in [64].

Given that the TOP is at least as hard to solve as the OP, heuristic approaches are the main solution methods found in the literature. The work presented in [75] uses a Tabu search heuristic to address the team orienteering problem. The proposed heuristic is embedded in an adaptive memory procedure which switches between small and large neighbourhood stages while improving the solution. Their approach uses both random and greedy methods for generating neighbourhood solutions. In addition, both feasible and infeasible solutions are explored. The proposed approach finds new better solutions for many of the benchmark instances in the literature while having smaller computational requirements.

A local search method for solving the TOP is presented in [76]. An algorithm combining various local search heuristics is presented. The use of *Guided Local Search* (GLS) is used to improve two of the presented heuristics. Extra heuristics are used to force the method to explore larger portions of the solution space. The proposed approach is compared with state of the art heuristics from the literature, and the results are of the same quality. In addition, the approach requires less computational time.

The work of *Vansteenwegen et al.* [77] uses iterated local search for solving the *Team Orienteering Problem with Time Windows* (TOPTW). In such a problem each location has not only a score but also a service time and a time window. The goal of this problem is to form maximising routes that visit a subset of the locations at the right time while respecting the given budget constraints. An example application is a personalised tourist guide that respects opening and closing times of attractions. The proposed approach is using insert operations combined with a shake operation that allows escaping local optima. Compared with the state of the art produces solutions with an average gap of only 1.8%, while for 31 instances, new best solutions are computed. In addition, it is fast enough to be applied online, as its average computation time is at least two orders of magnitude less than the state of the art.

An ACO approach for solving the TOP is presented in [78]. The authors present four different methods to generate candidate solutions. Specifically, sequential, deterministic-concurrent, random-concurrent and simultaneous methods are used. Their proposed approach is compared using standard benchmark instances from the literature finding better solutions than the state of the art in some of them.

Another work addressing the TOPTW problem is presented in [79]. The proposed solution uses a VNS approach operating on granular instead of complete neighbourhoods. This allows for increasing the efficiency of the algorithm without compromising its effectiveness. Compared to the state of the art it is a competitive method managing to improve 25 of the best-known solutions for benchmark instances. Another work that makes use of VNS is presented in [80] to tackle the Capacitated TOPTW. Their results are matching the state of the art in benchmark instances.

A simulated annealing approach is used in [81] to solve the TOPTW problem. They provide extensive computational results where the proposed approach performs on par with the state of the art. It manages to find a better solution to only a few benchmark instances, but it required more computational time.

A path relinking approach for the TOP is presented in [82]. Two variants of the approach are presented, namely a fast and a slow variant. They are tested against state of the art methods in benchmark instances from the literature. The fast variant achieves on average a 0.39% gap from the best-known solutions in

five seconds, while the slow variant manages to close the gap to 0.04% in 272.8 seconds. In addition, the slow variant manages to improve the best-known results for five benchmark instances.

3.2.3 Sampling and the orienteering problem

The tasks of sampling or information gathering have been modelled as instances of the OP in various works. In [83] a solution to the lake and river monitoring task for resource-limited robotic teams is presented. To achieve that an *efficient Single-robot Informative Path planning* (eSIP) approximation algorithm is proposed. This algorithm is used to optimise a path for a single robot. A Gaussian process is used to model the studied phenomenon. The amount of information collected by the planned path is calculated by using the Gaussian process and the mutual information between the visited points and the rest of the monitored area. It is shown that the eSIP gives close to optimal results. It is shown that the proposed approach is solving an instance of the *Submodular Orienteering Problem* (SOP) [84]. In addition, the proposed approach is extended to multi-robot teams using a sequential allocation method. The method is evaluated in field tests and using real-world sensor data sets.

The work presented in [85] addresses the additional information gathering problem in a sensor network. They present a path planning method for maximising mutual information. This work is an extension of [83] dealing with special cases for underwater vehicles. Close to optimal paths are produced using a recursive greedy algorithm. The generated paths take into account the lack of communication in underwater vehicles and the fact that the vehicles take measurements while moving. In addition, the path is guaranteed to avoid high-traffic areas that can pose a threat to the vehicle. The proposed approach is verified using field trials of an underwater glider.

Given the computational complexity of solving the submodular OP, it can be inapplicable for online usage on platforms with limited computational resources. In the work of *Heng et al.* [86] a linear approximation of the submodular OP is proposed for addressing this problem. The method is applied to a simultaneous exploration and coverage problem performed by a *micro aerial vehicle* (MAV). The proposed approach is used continuously online to plan a path for the MAV using

on-board limited computing resources. The algorithm aims at providing obstacle free and feasible paths that maximise the observations of unexplored space and the sensor coverage of the area given the sensor limitations. An example application is the automated 3D reconstruction of environments. The proposed approach is validated in simulation using a MAV equipped with a forward-looking depth camera. In terms of coverage, it outperforms an exploration-only approach while still managing to explore the entire environment.

An exact solution for the *informative path planning* (IPP) problem is presented in the work of *Binney et. al.* [19]. A Gaussian process is used to perform scalar field estimation. The IPP is formed as an optimisation problem that tries to maximise the variance reduction of the Gaussian process model. The optimisation problem is solved using a branch-and-bound method.

Another work addressing the marine sampling problem is presented in [87]. The authors try to find better ways than a classical "lawnmower" pattern to address the sampling problem. In this work three different methods are proposed, an optimised "lawnmower", a graph search algorithm based on A* and a GA based heuristic. They apply their approach to a phytoplankton monitoring mission. Results showed that the non-linear GA is trying to collect more data from high uncertainty areas, while the optimised "lawnmower" and the A* solving an OP instance maximised the area coverage.

In the work of *Tokekar et al.* [88] an agricultural monitoring application is presented. A multi-robot team composed by a ground vehicle and an aerial vehicle is tasked to estimate a nitrogen map of an area. The proposed solution consists of solving two different problems, one for each vehicle. For the ground vehicle a *sampling travelling salesperson problem with neighbourhoods* (SAMPLINGTSPN) is used for optimising the time-consuming task of collecting soil samples. For the areal vehicle, given its limited energy, an orienteering problem instance is used to maximise the aerial measurements. For solving the orienteering problem, a four-approximation algorithm [89] is used. This algorithm guarantees that the result will visit at least a quarter of the sampling points visited by an optimal approach. The proposed approach is validated using simulations and field experiments.

Many of the sampling optimisation approaches consider Euclidean distances between the sampling points. Unfortunately, paths generated by them are not feasible for vehicles having kinodynamic constraints. To address that problem the

work presented in [90], [91] proposes a VNS approach for solving the *Dubins Orienteering Problem* (DOP) and the *Dubins Orienteering Problem with Neighbourhoods* (DOPN). The proposed approaches successfully compute optimised paths, but their computational cost is prohibitive for online applications.

The work of [92] addresses the informative path planning problem in an online manner. The presented approach, named *Randomized Anytime Orienteering* (RAOr) aims at providing near-optimal solutions with runtimes that allow online applications. The algorithm combines a constraint satisfaction problem and a travelling salesman problem. This way it manages to restrict the search space and find good solutions in a reasonable amount of time. Experimental results using simulation and real-world scenarios show that they improve the runtime by an order of magnitude over the state of the art.

3.3 Summary

This chapter presented relevant literature regarding the MRTA and sampling under constraints problems. For the MRTA multiple approaches that address various levels of complexity are presented. Regarding the sampling problem, methods from the combinatorial optimisation field for one or more vehicles are presented, along with their application to the sampling problem. An in-depth description of optimisation theory and heuristics to the optimisation problem can be found in appendix A. The next chapter will present the system architecture used in the scope of this thesis.

Chapter 4

System

The previous chapter presented the relevant literature for the MRTA and the Sampling under budget constraints problems. This chapter aims at presenting the software stack used to provide a reliable method of information sharing among the robot team, as well as the way it was integrated with real robots for performing real-life experiments. A general view of the architecture can be seen in figure 4.1.

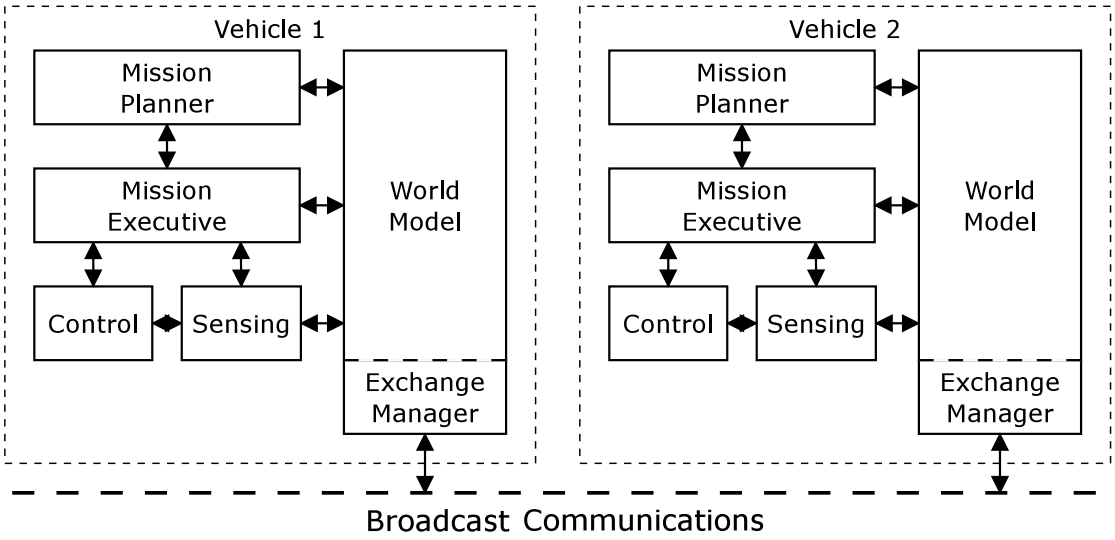


Figure 4.1. The software architecture used for evaluating the MRTA approach. The world model and the exchange manager are responsible for the reliable exchange of information among the members of the team. The mission planner implements the MRTA and mission optimisation. It uses the world model to access all the data required for the optimisation.

4.1 Reliable information sharing under communication constraints

As already mentioned in chapter 1, this work aims at providing an optimal MRTA approach that operates over latent and lossy communications. For that, an efficient and fault-tolerant way to exchange information among the team members is required. Such a method is presented in [12], where a *Distributed World Model* (DWM) architecture is described. This method was developed by members of the Ocean Systems Laboratory and was evaluated during field trials whose results are presented in [9].

The DWM architecture is an efficient and robust method to share information among the members of a robot team. It makes use of broadcast communications and aims at maintaining a consistent world knowledge state among the team members. This is achieved by the use of an ontology representation to store all the relevant information and a *Data Exchange Manager* (DXM) module that is responsible for sharing the data. These modules can be seen on the right side of each vehicle representation in figure 4.1.

The ontological data representation used by the DWM was first presented in the work of [93]. This work proposed the use of a temporal representation of knowledge. It is shown that such a representation is able to model better the evolution of information, as well as, the uncertainties of the environment that is modelled through the world model. An example of this representation applied to a target classification scenario can be seen in figure 4.2.

In this example, one can see the representation of a target that is discovered and needs to be classified. When it is first discovered at time 123, it has an *"Unknown"* class. After inspection and classification are performed, at time 145, the class is updated taking the value *"Rock"*. Likewise, one can model various aspects of the environment and the vehicle itself, as its position, intentions, health status, remaining energy etc. A more extensive work describing the use of ontologies in MCM missions can be found in [94]. The ontology framework used to model all these data is Jena developed by the Apache Foundation [95].

The second main component of the DWM is the DXM. The DXM is a software module that is responsible for sharing information with the other robots in a transparent manner to the rest of the software architecture. This means that the

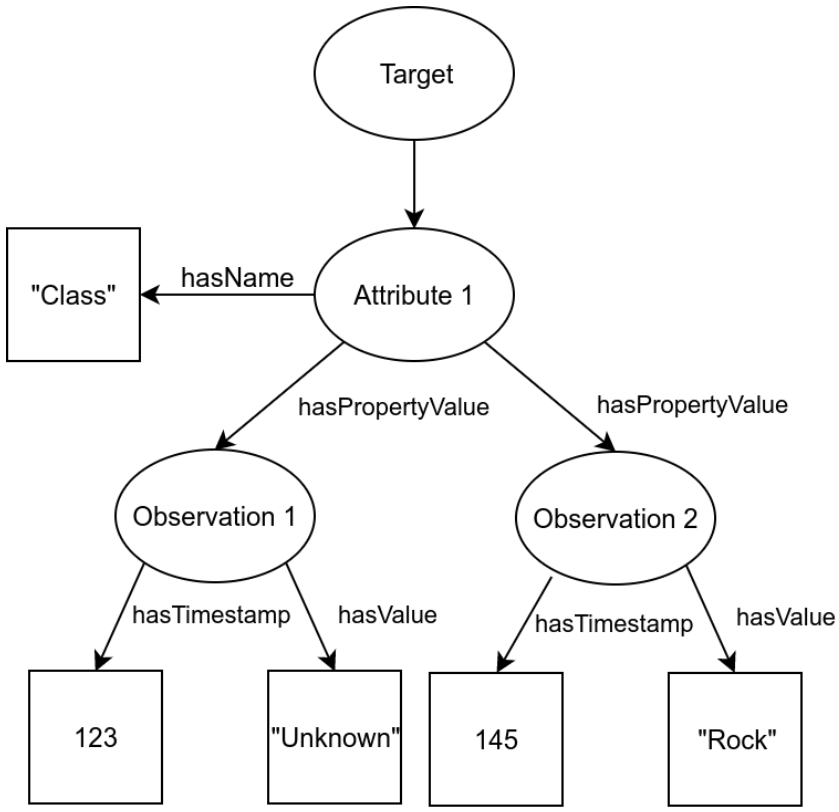


Figure 4.2. Example of temporal representation of knowledge for a target. Initially when the target is discovered at time 123 its' class is unknown. After the classification process finishes at time 145 it is classified as a "Rock".

planning software does not have to take into account any communication actions reducing the planning effort. The DXM is also tasked with receiving information from other robots, storing it in the ontology so that other software modules can access it, and finally acknowledging the reception. A flow chart of the DXM can be seen in figure 4.3.

As can be seen, the DXM is continuously running as a process. It checks if it is time to transmit any data, as the shared communication medium is accessed using a *Time-Division Multiple Access* (TDMA) policy. If it is time to transmit, it proceeds with the data selection and transmission. The data selection is based on the information needs of each vehicle and the maximum allowed packet size. The information needs of each vehicle are user-defined and are communicated at the beginning of the mission. After selecting the data, they are accordingly transmitted. On the other hand, if it is not the allowed time to transmit, the DXM checks if there is an incoming message. If it exists, then it unpacks the data and inserts them in the ontology. Following it arranges for the acknowledgement

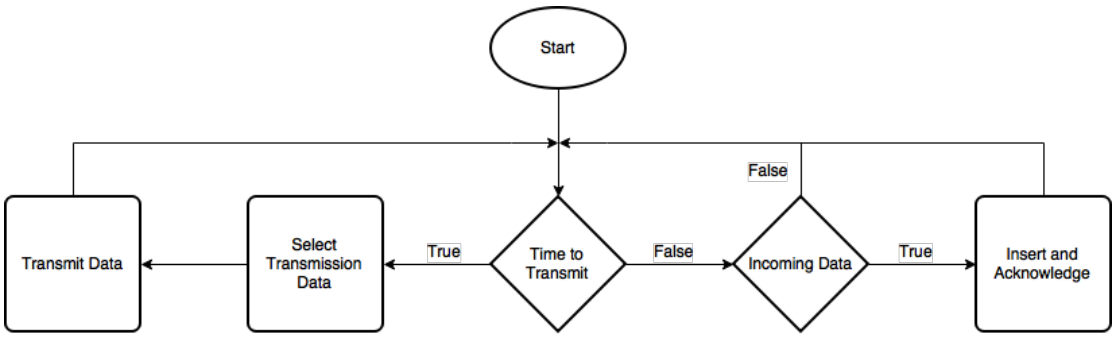


Figure 4.3. Execution diagram of the Data Exchange Manager. The DXM checks if it is time to transmit. If yes it selects the appropriate data that fit in a message based on their priority and accordingly transmits them. If not it is checking for any incoming data. If there are any, their reception is acknowledged.

of the message. For acknowledging messages, the DXM offers various policies that make it robust in high error rate environments. Details for the policies can be found in [12].

4.2 Integration on real platforms

The DWM architecture is primarily targeted to fleets of autonomous underwater vehicles communicating using acoustic modems. For testing with real robots in the scope of this work, additional software is written and integrated. The complete software stack used to run these experiments can be seen in figure 4.4.

Since the scope of this work aims at latent and error-prone communications, a communications simulator developed by the ocean systems laboratory is used. One of the primary considerations regarding the real platforms is how to integrate this communications simulator. The use of the simulator is imperative as it allows to adjust the channel parameters that affect the optimal MRTA. The decided solution is to have the higher level software of each platform run on a workstation along with the communications simulator while leaving just the lower level software on the onboard computer of the robots.

To achieve this physical separation a bridge is created. This bridge is responsible for transferring high-level commands, for example, navigate to point A, provide feedback from the action execution and allow queries to the lower subsystems to be performed. This bridge is deployed as a ROS node that is translating from

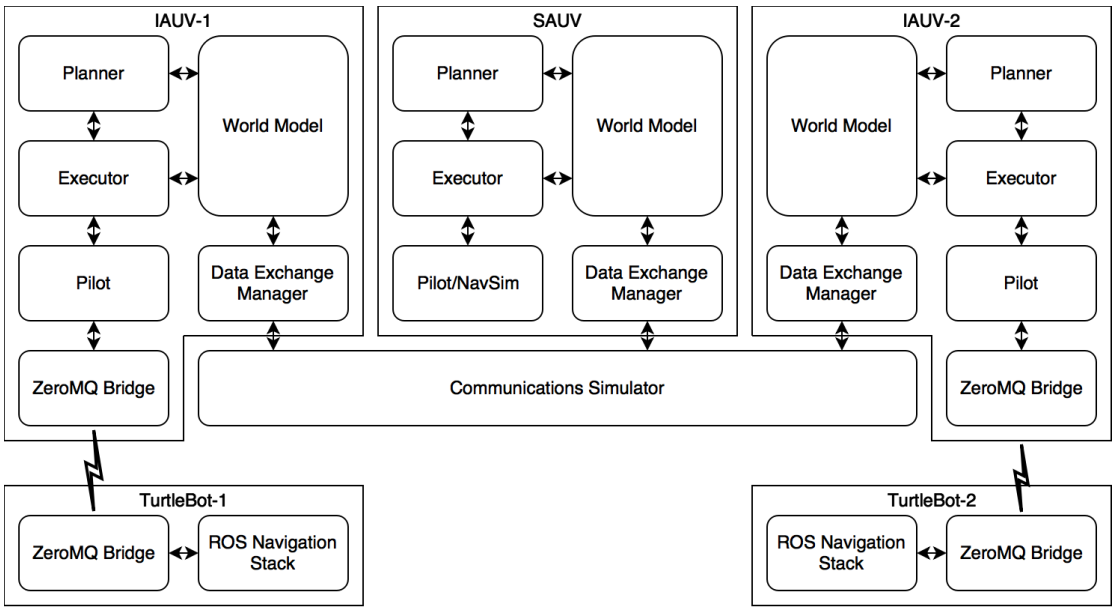


Figure 4.4. Software setup for the integration of Turtlebots with the OSL software architecture. The high-level software, as well as the communications simulator, is running on a workstation. On each robot, the ROS navigation stack is running providing navigation capabilities. Communication between the high-level software and the robots was performed using a ZeroMQ based bridge operating over WiFi.

various topics to ZeroMQ [96] publishers and subscribers. The ZeroMQ is used as a transport layer over WiFi between the robots and the workstation. On the workstation, the bridge node is translating high-level navigation commands and sending them to the low-level controls, while receiving feedback on the navigation execution. It also issues queries to the low-level path planners regarding the cost of various paths. Equivalently, the bridge node on the robot is receiving the requests and commands and, after consulting with the low-level ROS navigation stack, is reporting back to the high level.

4.3 Summary

This chapter presented the software architecture used to provide a reliable method of information sharing among the robot team. In addition, it presented the software stack used for integrating a communications simulator with the DWM and real robots. These developments were used to evaluate the proposed MRTA approach described in the next chapter.

Chapter 5

Multi-robot task allocation under communication constraints

The previous chapter presented an architecture for reliably transmitting information over channels that have high latency and errors in communication. In addition, it presented how it is integrated with indoors robotic platforms that facilitate the performance of experiments on real robots.

This chapter presents the proposed approach for addressing the MRTA problem under communication constraints using the aforementioned architecture. In particular, it studies how the MRTA can be solved optimally for a robotic team performing inspection tasks in conditions where communications are limited and latent. In addition, it studies the effects of communication errors in the MRTA problem solution. An example domain where communications are rather limited is the underwater one. There, acoustic communications, implemented using acoustic modems such as the ones presented in [97], is the primary way of communication as electromagnetic radiation is propagating poorly in water.

In the beginning, a typical structure of an underwater inspection mission will be presented, along with example applications. Following, the problem is analysed and classified according to the theory presented in chapter 2. Next, the MRTA is modelled as an optimisation problem. In addition, the data requirements of this formulation are discussed, and their integration within the DWM architecture is presented. Finally, the method is tested in simulation and using real robots under various levels of communication errors. Its performance is compared with other baseline methods, and the results are discussed.

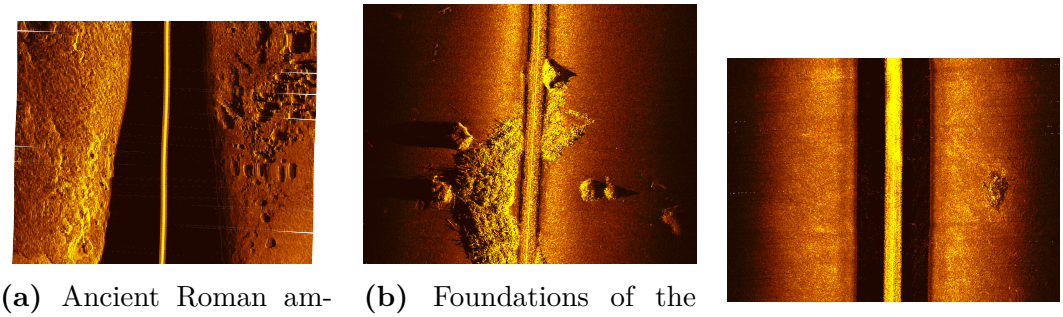
5.1 Underwater inspection mission

Underwater inspection missions usually involve one or more vehicles trying to locate objects of interest and collect information about them. A typical example of such a mission is the of *Mine-Countermeasures* (MCM) performed autonomously. It is one scenario that receives lots of attention in the literature. According to [98], MCM is the phase where more effort is spent in naval countermine warfare.

A typical autonomous MCM mission consists of three phases, each one requiring different capabilities [99], [100]. The first phase is called *Search-Classify-Map* (SCM) and aims at mapping an area, searching for potential mine objects and classifying them as *mine-like contacts* (MILCOs) or *non-mine mine-like objects* (NOMBOs). For this phase, usually, a large area needs to be searched. This is performed using a fast moving vehicle, equipped with appropriate sensors such as *side-scan-sonar* (SSS), that can collect data over the required area in a fast manner. The second phase requires the MILCOs to be revisited for the final identification as a mine and is named *Reacquire-Identify* (RI). Each of the MILCOs has to be localised based on the findings of the previous phase and appropriately inspected. For the inspection, the object must be viewed from different viewpoints. The final phase is referring to the neutralisation of the mine. If a mine is detected and identified then a vehicle is used to neutralise it. This is performed by either placing an explosive charge next to the mine and detonating it, or by directly guiding a disposable vehicle on the mine.

The phases of an MCM mission can be directly applied to other fields. For example, in underwater archaeology [101], a survey of the area of interest is required to create a map and locate potential archaeological artefacts. Then, any findings are closely inspected and photographed so that a 3D reconstruction of the object can be made. Examples of findings from such missions can be seen in figures 5.1 and 5.2.

During the ARROWS project, several field expeditions took part. Some results from search missions conducted autonomously using an IVER-3 vehicle are presented here. The first set presents artefacts with archaeological significance captured in Trapani, Sicily and Tallinn, Estonia. In figure 5.1a at the right part one can see the load of amphoras of an ancient Roman ship found off the coast of Trapani in Sicily. Figure 5.1b shows the foundation of the citadel and its towers



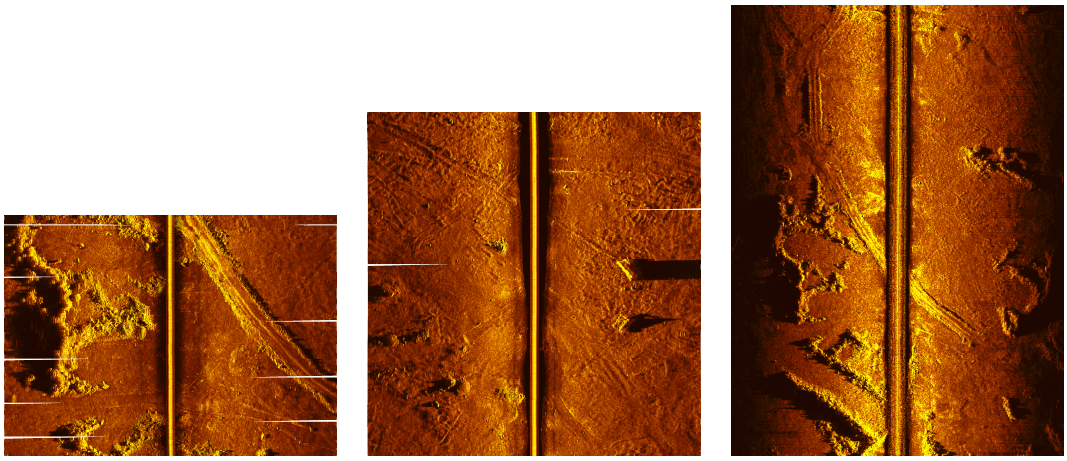
(a) Ancient Roman amphorae sitting on the bottom off the coast of Trapani, Sicily.

(b) Foundations of the citadel and its towers at the port of Tallinn, Estonia.

(c) Shipwreck off the coast of Naissaar island in Estonia.

Figure 5.1. Side-scan sonar images of antiquities captured during trials for the ARROWS project.

that were built to protect the harbour of Tallinn in Estonia. In this mission, the vehicle passed directly above the structure. The image is shown in figure 5.1c shows the shipwreck of a ship found just off the coast of the island of Naissaar in Estonia.



(a) RUMMU-1 **(b)** RUMMU-2 **(c)** RUMMU-3

Figure 5.2. Side-scan sonar images of machinery and structures captured during trials for the ARROWS project.

The second set of pictures was captured in lake Rummu in Estonia. The lake was formed in the place of a limestone quarry after the activity was halted and groundwater stopped being pumped out of it. During the flooding, several utility buildings and machinery were left in place and can be seen. In figure 5.2a one can see walls of structures on the left part of the image and part of a road on the right part. The second figure, shown in 5.2b shows two pieces of machinery on

the right-hand side of the image. Finally, in figure 5.2c a larger area was mapped showing structures, roads and other environmental features.

5.2 MRTA Problem

As already described before by the example inspection missions, the various phases of the applications require different capabilities. All these capabilities can be provided by a single vehicle. This vehicle would have to be equipped with a plethora of sensors, as well as, have enough energy and processing power to accomplish the mission. All these factors would make the vehicle big enough to be hard to operate without a large accompanying service ship. Additionally, this vehicle would be a single point of failure for the mission. In case of some error on it, the mission would not be able to be performed. Moreover, a single vehicle would require more time to perform the whole mission, raising the cost and the required effort.

Given the discussed disadvantages, this work focuses on solving the problem using specialised vehicles for each role. These vehicles can be simpler and smaller. Reducing the costs for acquiring, as well as operating, since they can be deployed by a small team using a small boat or even from the shore. In addition, a multi-vehicle team achieves task parallelisation which reduces the mission execution time. Furthermore, the use of multiple vehicles provides higher execution robustness as single vehicles are allowed to fail and still the mission will be able to be completed by the rest of the fleet. Following the different types of vehicles are described.

The first role is of a fast vehicle equipped with sensors that can search and map a large area. This vehicle will be referred to as *Search Autonomous Underwater Vehicle* (SAUV). Its' aim is to find potential targets so that they can be inspected. One example is IVER3 of OSL equipped with a Klein 3500 side-scan sonar which can be seen in figure 5.3. The second role will be fulfilled by a slower but hover capable vehicle that can stay close to a target and capture data with its' sensors from various viewpoints. Sensors that can be used in that case include cameras and forward-looking sonar. This type of vehicle will be addressed as *Inspection Autonomous Underwater Vehicle* (IAUV). Nessie AUV developed by the OSL can be used to fulfil that role and can be seen in figure 5.4.

The goal of the research presented here is to perform such missions in an

Figure 5.3. The IVER-3 AUV of the Ocean Systems Laboratory manufactured by OceanServer. It is equipped with a Klein-3500 sidescan sonar. This vehicle is not hover capable and requires to be in constant motion to be able to be submerged. It was used to acquire the data shown in figures 5.1 and 5.2.

optimised way using a fleet of collaborating vehicles. These vehicles operate simultaneously, and the mission evolves in an online manner. Specifically, inspection targets are gradually discovered and have to be inspected in an optimised way. To achieve that the vehicles can share information and cooperate using underwater acoustic communication. The main problem to be addressed is which the inspection candidates are visited by which vehicle. A formal definition of the problem will be presented in the next sections.

5.2.1 Task description

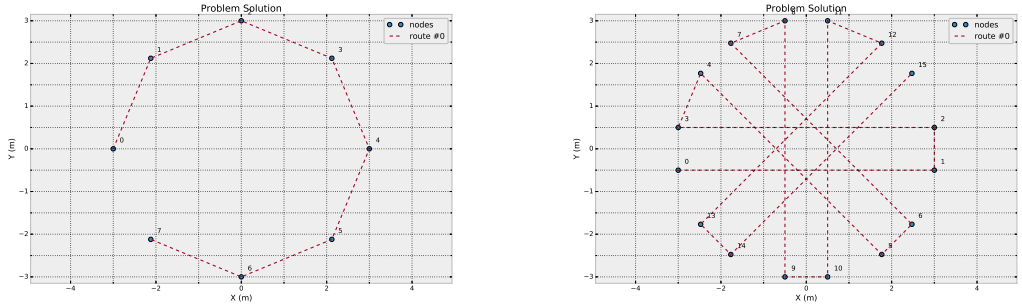
As briefly described the tasks for the inspection vehicles require to collect data from various points so that the object can be correctly classified. Given the task description, it is easy to see that it can be decomposed into several simple tasks each one being a data collection action from a specific point. Such simple tasks can be allocated to different agents. Additionally, such a task can have multiple

Figure 5.4. The Nessie-VII AUV developed by the Ocean Systems Laboratory. It is hover capable and has five degrees of freedom. It is equipped with a P900 forward-looking multibeam imaging sonar manufactured by BlueView.

decompositions depending on the kinematic capabilities of the vehicles performing the task. Two different decompositions can be seen in figures 5.5a and 5.5b.

The first decomposition, seen in figure 5.5a, refers to a hover-capable vehicle, equipped with a forward-looking sensor. The task is decomposed in a set of simple tasks, presented as blue dots, that form a circular pattern around the object to be inspected. At each task, the vehicle has to stop and capture data using its sensor. If executed by a single vehicle, it has to move in a circular pattern through the waypoints as shown by the path in figure 5.5a.

The second decomposition refers to a non-hover capable vehicle, equipped with side-scan sonar sensors and can be seen in figure 5.5b. This type of vehicle requires to be in constant motion in order to remain submerged. In addition its' sensor configuration allows reliable data to be captured only when collected during straight paths. These requirements lead to simple tasks of traversal lines around the object of interest. For example, such a line is denoted by waypoints 0 and 1



(a) Inspection pattern of a hovering capable vehicle, equipped with a forward looking sensor.

(b) Inspection pattern of a non-hover capable vehicle, equipped with sidescan sonar sensors.

Figure 5.5. Inspection patterns for vehicles with different kinematics.

in figure 5.5b.

According to the task description presented in section 2.1.1 the studied inspection task can be classified as a complex task since it has multiple decompositions based on the vehicle type. In the scope of this work, inspection is assumed to be performed only using hover capable vehicles. In addition, an inspection of an object must be performed by a single vehicle. Such a requirement arises from the fact that the object needs to be classified online during the mission execution by one of the inspection vehicles. Given the limitations in communication bandwidth in the studied scenario, it is prohibitive to communicate sensor readings between vehicles. These considerations allow treating the tasks to be allocated as decomposable simple tasks since there is a single way to decompose them and they can be allocated to a single agent only.

5.2.2 Utility description

As discussed in section 2.1.3 each MRTA can be classified based on the agent-task utilities. The utility for an agent performing a task is shown in (2.2). Where the utility is defined as the difference between the quality of performing a task Q and the cost for performing a task C. For a simple task, it can be assumed that the quality of the task Q is the sum of the qualities of all the n simple tasks q composing the decomposable simple task as shown in (5.1).

$$Q = \sum_{i=1}^{n} q_i \tag{5.1}$$

Applied to the inspection scenario, the utility for performing an inspection can be assumed to be constant and equal to the sum of the utilities for getting readings from various viewpoints. Regarding the cost of performing the decomposable simple inspection task, it can be represented by the cost of travelling to the object that needs to be inspected plus the cost of moving around performing the n different simple tasks as it is shown in (5.2).

$$C = c_t + \sum_{i=1}^{n} c_i \tag{5.2}$$

Since the inspection pattern around each target is the same the cost of performing the data collection can be considered constant and equal to the sum of the costs performing the n simple tasks as described by the term $\sum_{i=1}^{n} c_i$. Thus, the cost of each decomposable simple inspection depends only on the cost to travel to each object that needs to be inspected c_t. It is easy to see that this cost is dependent on the schedule of inspections an agent has to perform. In other words, the relative ordering of the targets can change the cost of travel for each one of them. Such a behaviour is observed in MRTA problems that show *in-schedule dependencies* (ID) as shown in section 2.1.3 and figure 2.3.

5.2.3 MRTA as an optimisation problem

The previous section showed that the MRTA problem for underwater inspection tasks shows in-schedule dependencies based on the cost of travel between the various inspection targets for each vehicle. Such problems can be modelled as optimisation problems minimising the travelling costs of the robots in the team.

To achieve that, the problem can be modelled as a fully connected undirected graph that contains a vertex for each target to be inspected and a vertex for the starting position of the agents. The weights on the edges connecting the vertices represent the cost of travelling between these two vertices. The optimisation procedure must, therefore, find m subsets of vertices, each containing the starting

vertex. In each subset, a Hamiltonian cycle is formed, and the sum of their costs must be minimised. Such a problem is named as *multiple Travelling Salesmen Problem* (mTSP) in the literature. An mTSP problem can be formulated as a *Mixed-Integer Linear Program* (MILP) as presented in [29] and can be seen below.

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5.3}$$

$$s.t. \sum_{j=2}^{n} x_{1j} = m, \tag{5.4}$$

$$\sum_{j=2}^{n} x_{j1} = m, \tag{5.5}$$

$$\sum_{i=1}^{n} x_{ij} = 1, \qquad j = 2, ..., n \tag{5.6}$$

$$\sum_{j=1}^{n} x_{ij} = 1, \qquad i = 2, ..., n \tag{5.7}$$

$$u_i + (L-2)x_{1i} - x_{i1} \leq L-1, \qquad i = 2, ..., n \tag{5.8}$$

$$u_i + x_{1i} + (2+K)x_{i1} \geq 2, \qquad i = 2, ..., n \tag{5.9}$$

$$x_{1i} + x_{i1} \leq 1, \qquad i = 2, ..., n \tag{5.10}$$

$$u_i - u_j + Lx_{ij} + (L-2)x_{ji} \leq L-1, \qquad 2 \leq i \neq j \leq n \tag{5.11}$$

$$x_{ij} \in \{0, 1\}, \forall (i,j) \in A. \tag{5.12}$$

The objective function to be minimised is shown in (5.3). The sum describes the total cost of traversing all the vertices. The cost of travelling from vertex i to vertex j is given by c_{ij} and x_{ij} is a binary variable denoting the existence of a path between the two vertices in a solution. A is the set of arcs that connect the vertices.

Constraints (5.4) and (5.5) are used to ensure that the number of routes created are equal to the number of salesmen m. Appart from the starting vertex, all the other vertices are allowed to be visited only once. This can be achieved by using constraints (5.6) and (5.7).

In this formulation, the minimum and the maximum number of vertices a salesperson can visit can be set by variables K and L respectively. Constraints (5.8) and (5.9) enforce that these settings are respected. It should be noted that

the formulation is valid only for $2 \leq K \leq \lfloor (n-1)/m \rfloor$ and $L \geq K$, where n is the number of vertices and m is the number of salesmen.

Constraint (5.10) prevents the salesmen to travel to only one vertex and is required only in cases where $K < 4$. Finally, constraint (5.11) is a sub-tour elimination constraint preventing the creation of disconnected sub-tours. A solution of a simple mTSP problem can be seen in figure 5.6.

This method can be used in cases where multiple vehicles are deployed to and extracted from the same place. It optimises the distance each vehicle has to travel while respecting the minimum and maximum inspection targets that need to be visited. Tuning the minimum and maximum targets one can affect the utilisation of the team.

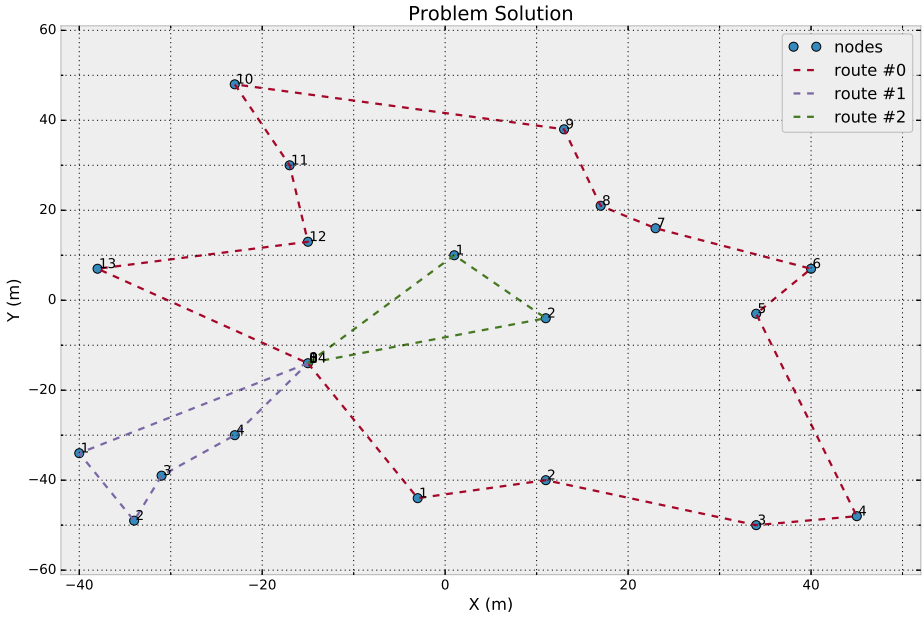


Figure 5.6. Solution to an MTSP instance of 20 randomly generated points using three salespeople. The first vertex was selected to be as the starting point where all the tours start from and finish to. The minimum number of visited vertices per tour K is 2, while the maximum L is 20. The total cost of the tours is 476.444.

Given that in the examined scenario inspection targets are dynamically discovered during the mission execution one should take into account the position of the robots at the time that the problem is solved. This can be modelled by adding m vertices to the graph each one representing the current position of the robot.

The solved problem then it becomes the *Multidepot multiple Travelling Salesmen Problem* (MmTSP). A special case of this problem is when all the agents start from different locations, but they must end their trip on a single final location. The *Multiple departures single destination mTSP* (MDmTSP) problem is of particular interest for the underwater inspection mission MRTA as it is desirable to have all the vehicles end their mission on the same place to be collected. The work presented in [102] presents a MILP formulation for the MDmTSP that is detailed below.

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{5.13}$$

$$s.t. \sum_{j\in V'} x_{ij} = 1, \qquad i \in D \setminus \{1\} \tag{5.14}$$

$$\sum_{i\in V'} x_{ij} = 0, \qquad j \in D \setminus \{1\} \tag{5.15}$$

$$\sum_{j\in V'} x_{1j} = 0 \tag{5.16}$$

$$\sum_{i\in V'} x_{i1} = n \tag{5.17}$$

$$\sum_{i\in V} x_{ij} = 1, \qquad j \in V' \tag{5.18}$$

$$\sum_{j\in V} x_{ij} = 1, \qquad i \in V' \tag{5.19}$$

$$u_i + (L-2)\sum_{k\in D} x_{ki} - \sum_{k\in D} x_{ik} \leq L-1, \qquad i \in V' \tag{5.20}$$

$$u_i + \sum_{k\in D} x_{ki} + (2+K)\sum_{k\in D} x_{ik} \geq 2, \qquad i \in V' \tag{5.21}$$

$$x_{ki} + x_{ik} \leq 1, \qquad k \in D, i \in V' \tag{5.22}$$

$$u_i - u_j + Lx_{ij} + (L-2)x_{ji} \leq L-1, \qquad i \neq j, i, j \in V' \tag{5.23}$$

$$x_{ij} \in \{0,1\}, \forall i,j \in V \tag{5.24}$$

The objective function of the MDmTSP is shown in (5.13). As in the previous problems, c_{ij} describes the cost of travel from vertex i to vertex j while x_{ij} is a binary variable denoting that a path from i to j exists in the solution. Constraints (5.14) and (5.15) ensure that from each depot other than the sink depot one salesman exits while none returns. Accordingly, constraints (5.16) and (5.17) enforce

all the n salesmen to finish their tours in the sink depot, while no tour is permitted to start from there. All the other vertices are allowed to be visited by only one salesman. This is shown by constraints (5.18) and (5.19). Constraints (5.20) and (5.21) set the minimum and maximum vertices each salesman is allowed to visit. In constraint (5.22) the salesman is prohibited to go directly from his starting depot to the ending depot. Finally, constraint (5.23) is a sub-tour elimination constraint.

In figures 5.7 and 5.8 the solution of two MDmTSP instances are presented. In the first case, the salesmen are allowed to visit the minimum amount of vertices if it is viable producing a minimal overall distance travelled for all the agents. In the second case, each salesperson is forced to traverse at least five vertices other than the starting and ending depot. That, as a consequence, affects the overall solution cost. These parameters are tuned using the variables K and L.

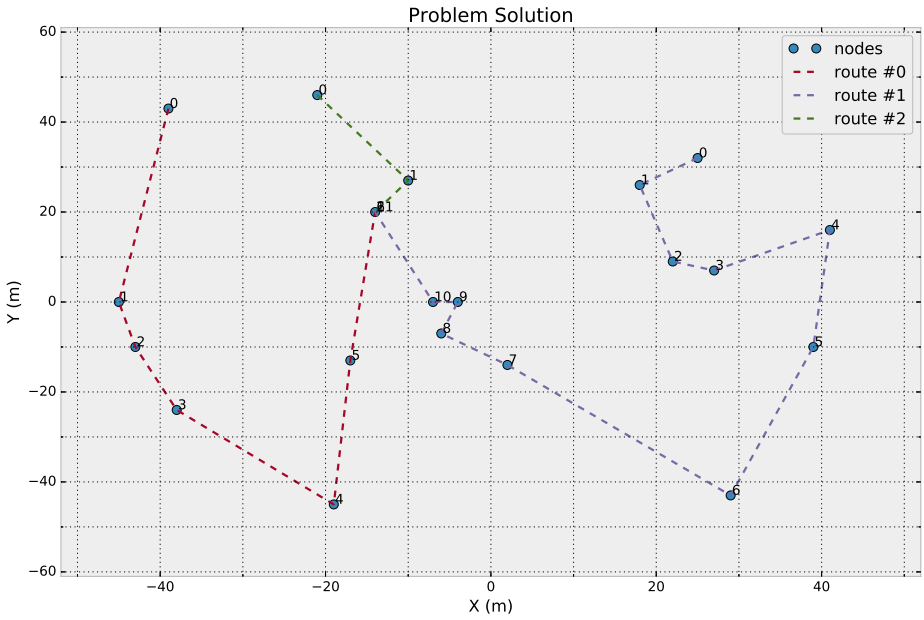


Figure 5.7. Solution to an MDmTSP instance for three agents. 16 randomly generated vertices had to be visited. Three randomly generated vertices denoted the start positions of the three agents. A randomly generated vertex was the ending point of the tours. The total cost of the tours is 383.010.

For the inspection scenario, this can represent vehicles starting from different positions in the world. In addition, this formulation can be useful when there is a need to replan a schedule after the mission has started. It will be able to capture

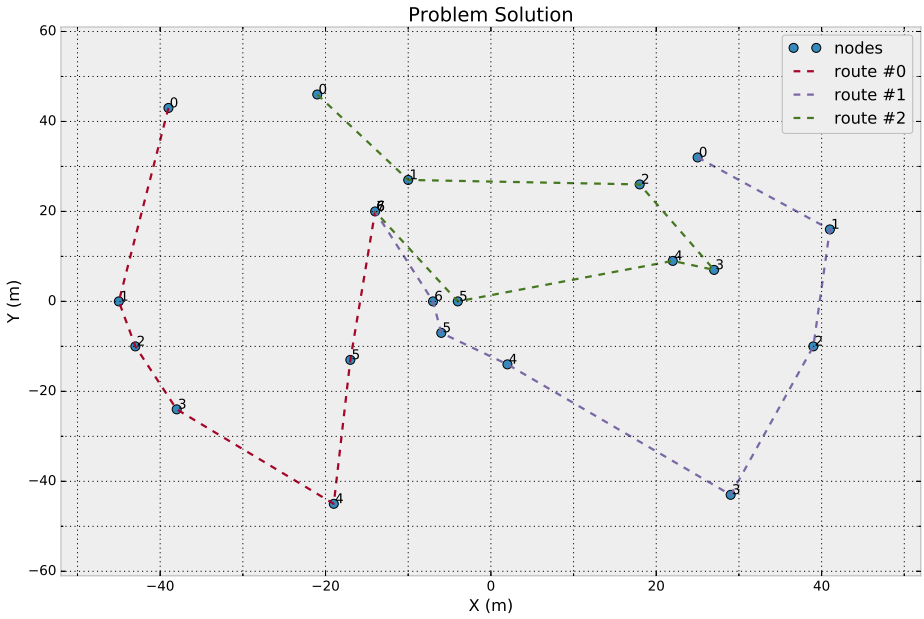


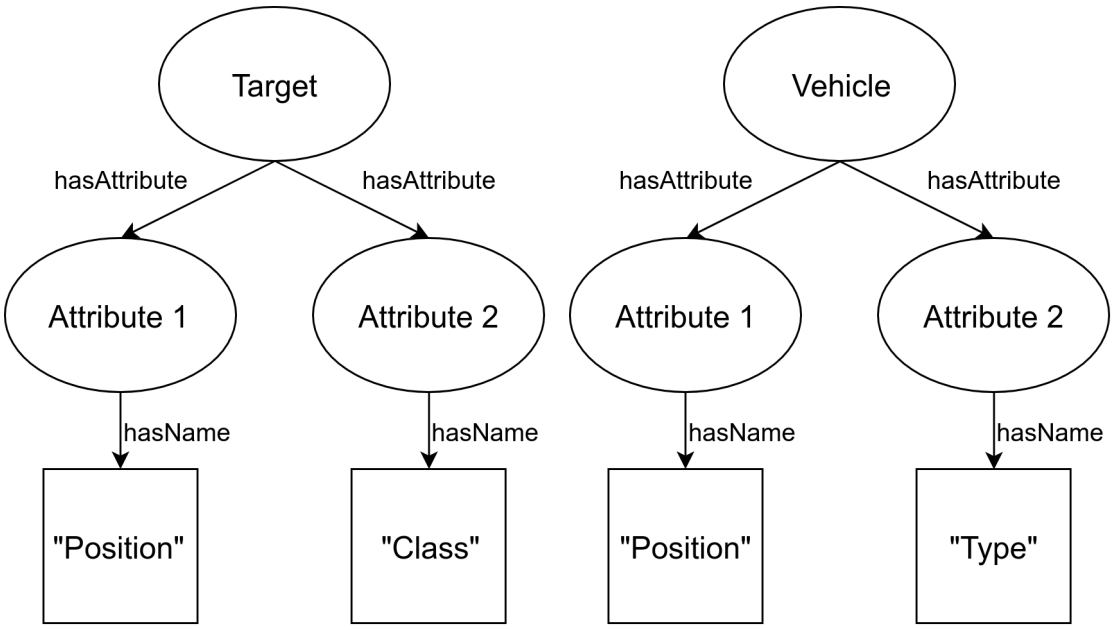
Figure 5.8. Solution of an MDmTSP instance with minimum visits per agent. The solution takes place on the same vertices as in 5.7. Each agent is required to visit at least five vertices other than its starting vertex and the finishing vertex. The total cost of the tours is 449.955.

the dynamics of the mission with the vehicles being in different positions, as they are performing different tasks.

5.2.4 Data modelling

As described in the previous section, the optimisation procedure constructs an undirected graph containing the positions of the vehicles, the positions of objects to be inspected as well as the final position where the vehicles must end their mission. It can be seen that information regarding the vehicles as well as the inspection targets is vital for task allocation. The semantic representation of the required data stored in the DWM architecture can be seen in figure 5.9.

Regarding the information priorities that the DWM supports it was decided to give a higher priority to the target information. It is more important to notify the vehicles of the team when a target is discovered or classified as this will trigger a new task allocation round. The vehicle position can be accurate enough for task allocation even if it has diverged a bit since the last time that information was received.



(a) Representation of a target in the DWM.

(b) Representation of a vehicle in the DWM.

Figure 5.9. DWM architecture representation of the data required for performing task allocation.

5.3 Experimental setup

As described before an inspection mission requires the team to search an area for potential targets that need to be inspected and classified. At the beginning of the mission, none of the targets is known. The targets are incrementally discovered as the mission progresses, and are then inspected and classified. In this work, the new targets are discovered at a user-defined interval. As soon as an inspection target is discovered, its information is shared with the other team members. They then decide on a task allocation for all the currently not inspected targets. The task allocation is performed in a distributed manner. Each robot will calculate a task allocation solution based on its local knowledge about the team status and execute the part that is allocated to itself. The fact that the communications are erroneous can lead to inconsistencies of the knowledge between the agents. This can lead to different allocations and thus different performance.

For the mission studied in the scope of this work the team is composed of a single SAUV, designated to search an area for targets, and two IAUVs, that are inspecting those targets. For the simulation experiments, the vehicles are

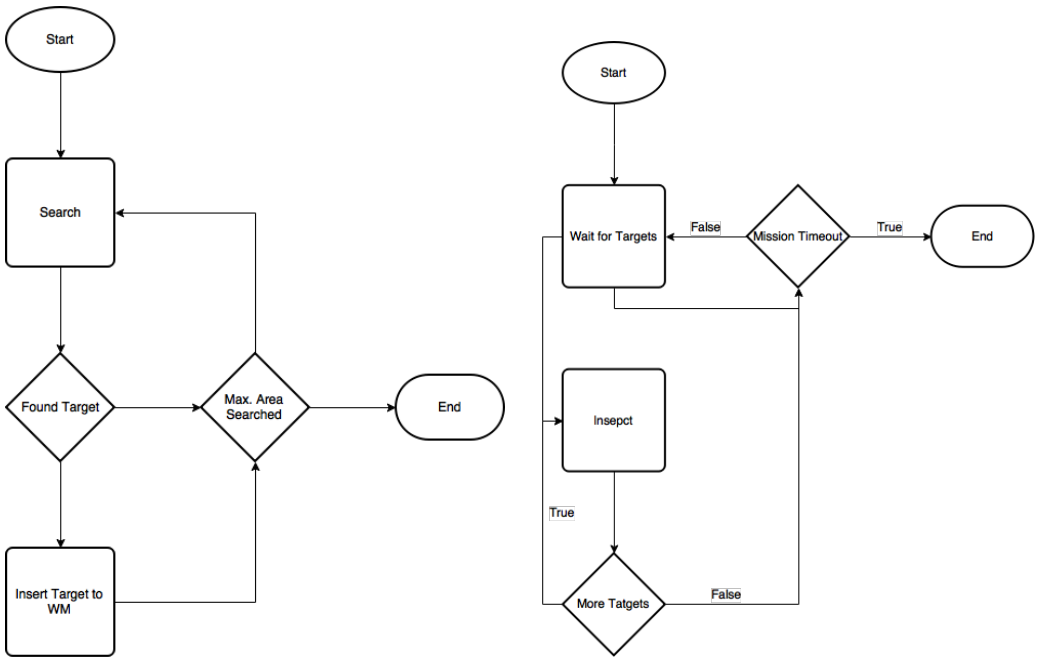
simulated using the navigation simulator for the Nessie VII AUV [103]. This simulator is a dynamics simulator that uses the hydrodynamic model of the AUV and is developed by the Ocean Systems Laboratory. In the real world experiments, the SAUV is still simulated, while the IAUVs are represented by two TurtleBot2 robots [104].

Regarding the communication among the team, it should resemble the underwater channel, which is error-prone, has low bandwidth and has high latency. For that, a communications simulator is used, that was developed by the Ocean Systems Laboratory. The communications simulator is an application level simulator that allows user-defined error, bandwidth and latency level in communications.

To execute its role in the mission each of the vehicles has modules that are responsible for planning its actions and executing those plans. Those are the software parts that can be seen in the top left corner of figure 4.1.

The planning module is present in the IAUVs and is the one that is responsible for running all the optimisation algorithms that were already described in the previous section. This module consults the world model through a client library and generates plans to be executed. Whenever new information regarding the mission is received through the world model, the module devises an updated plan and submits it to the executor module. The executor module flowchart for the two types of vehicles can be seen in figure 5.10.

Figure 5.10a shows the state machine of the search vehicle. It starts its mission searching for targets. Whenever a target is found it is inserted in the world model. The world model, in turn, is responsible for synchronising this information with the inspection vehicles, as has been already mentioned. When the maximum area is reached the SAUV finishes its mission and goes to the extraction point. The state machine of the IAUVs is respectively shown in figure 5.10b. The vehicle stays in place waiting for a list of targets that has to inspect from the planner. As soon as it receives one, it starts executing the mission. It iteratively follows the list of targets to inspect until the maximum mission execution time is reached when it moves to the extraction point. Whenever an updated list of targets is received the planner triggers a reset in the state machine of the IAUV unless it is already inspecting a target.



(a) State machine for an SAUV. **(b)** State machine for an IAUV.

Figure 5.10. Execution state machines for the two types of AUVs taking part in the mission. The SAUV is searching for targets until a maximum search area is covered. Whenever a target is found it is inserted in the world model. The IAUV is waiting for targets until a maximum mission time is reached. Whenever a target is received it is inspected.

5.4 Evaluation algorithms

To evaluate the performance of the proposed optimisation based task allocation methods, they are compared against two different allocation schemes.

The first allocation scheme is a greedy one. In that scheme, the targets are assigned taking into consideration only the distance of an inspection target from each candidate AUV. Whenever a new target is discovered the distance from each robot is calculated. Then the target is assigned to the robot currently closest to the target. It should be mentioned that in the current implementation of the greedy algorithm once a target is assigned to a vehicle, it cannot be reassigned to any other at a later time.

The second allocation strategy is based on the k-means algorithm as presented in [105]. This allocation strategy tries to take advantage of the spatial proximity of the targets to be inspected. The k-means algorithm, as presented in [106], provides a method of clustering a set of observations based on a distance metric. Given a set of n observations $X \in \mathbb{R}^d$, the algorithm tries to partition these observations into k sets with $k \leq n$. It is achieved by choosing k mid-points M that minimize the potential function shown in equation (5.25).

$$\phi = \sum_{x \in X} \min_{m \in M} \|x - m\|^2 \tag{5.25}$$

Algorithm 5.1 k-Means clustering

Input: Number clusters k, List of points
Output: Cluster centres

```
1: for i = 1 to k do
2:     m_i = GetRandomCentre
3: while M changes do
4:     for i = 1 to k do
5:         GetClosestPoints(i)
6:     for i = 1 to k do
7:         UpdateCentreOfMass(i)
8: return m
```

The procedure is simple and can be seen in algorithm 5.1. Initially, the algorithm chooses k random centres, one for each cluster. Then it calculates the points that are closer to centre m_i than any other centre and assigns them to cluster M_i.

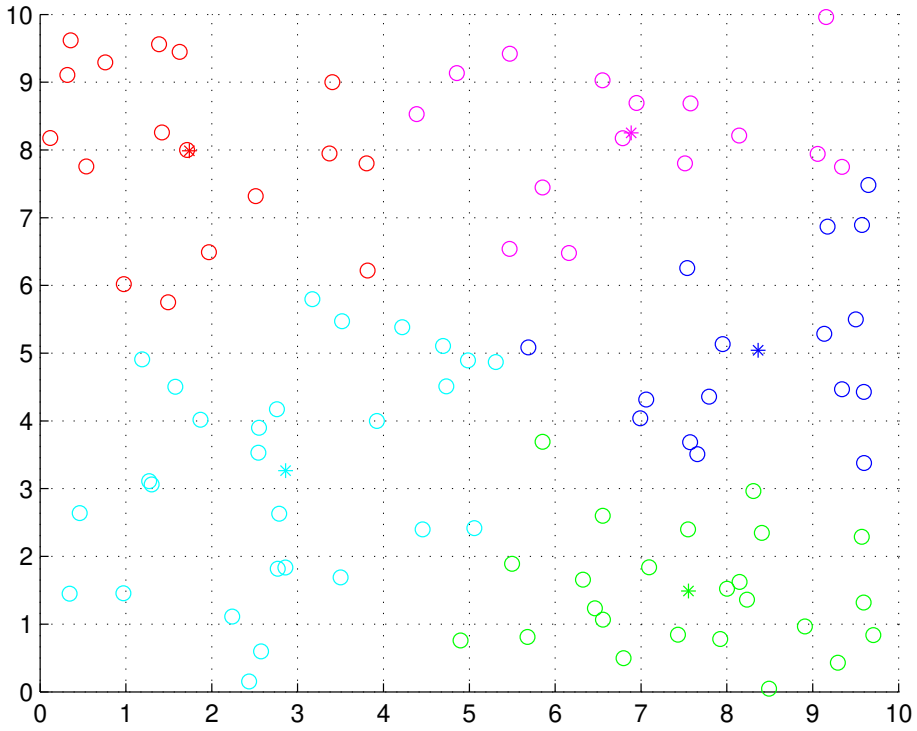


Figure 5.11. Clustering of 100 2-d observations in five clusters using the k-means algorithm. The stars represent the k centres and the data points are coloured the same as their respective centre of mass.

Following that, it calculates a new centre of mass for each cluster based on the points assigned to that cluster. This process is repeated until all the centres of mass are not changing. An example of the k-means algorithm can be seen in figure 5.11. In this paper, the k-means++ algorithm is used, as presented in [107].

In the case of task allocation studied in this work k is set to be equal to the number of IAUVs, thus creating one cluster per IAUV. Whenever a new target is detected the clustering algorithm clusters all the unclassified targets. Each cluster is then assigned to the IAUV that is closest to the mid-point of the cluster. Finally, each IAUV solves a fast TSP instance to obtain the optimal way to visit the targets within a cluster.

5.5 Results

The previous section described the different aspects of the experimental setup used to collect results on the proposed MRTA methods. This section will present the results obtained through simulations and real robot experiments.

5.5.1 Simulation results

The first set of results were collected through simulations and were presented in [108]. The robots had to perform a simulated inspection mission, like the ones described before. This work evaluated the performance of the mTSP optimal allocation method against a greedy and a k-Means based heuristic.

The mission starts with the three vehicles placed close to each other and having no knowledge of any targets. Then the SAUV starts discovering a new, randomly generated, target every minute until the total amount of discovered targets is reached. For these simulations, the number of targets to be discovered was set to ten. The time interval between the discovered targets is chosen so that the target discovery is spread during the mission. A lower interval would be equivalent of knowing all the targets at the beginning of the mission, while a much larger interval would have only one task allocated each time defying the purpose of needing an MRTA strategy. The amount of targets is chosen so that the mission is not too long while being long enough for requiring some more sophisticated MRTA.

As new targets are discovered, they are inserted in the world model of the SAUV and are respectively synchronised with the world models of the IAUVs. Whenever an IAUV receives a target, it allocates it to one vehicle according to its allocation strategy. It then proceeds to execute its updated plan. To simulate the inspection and classification phases for the IAUVs, an inactivity period of 1.5 minutes is imposed whenever they reach a target. This is chosen again for allowing enough targets to remain unclassified and the MRTA to be required. Finally, the mission ends when all the targets have been classified and the classification information propagated to the whole team.

To ensure the correct evaluation of the allocation methods a systematic approach has to be followed. To avoid favouring a specific method due to a specific target generation pattern, ten sets with randomly generated targets are used. The targets are randomly generated in an area of 100 by 100 meters around the initial

position of the SAUV. The generation is performed offline, and the targets are provided to the SAUV at the beginning of the mission. It then iteratively inserts them in its world model at the predefined rate.

To evaluate the effect the underwater communications channel has on the efficiency of the different task optimisation strategies, various communication conditions are tested. Initially, the maximum size of each acoustic packet is set to 512 bytes. This is a typical value for some of the underwater modems available. The time it takes to travel from the source to the destination is set to two seconds. This value is chosen given the sound propagation speed, the average distance of the vehicles during the mission and the time to encode and decode the information from the modem. The time slot duration for each vehicle, for the TDMA MAC control of the channel, is set to ten seconds. Finally, different *packet error rates* (PER) are used. This is done to show how the dropping quality in the communications channel affected the mission planning execution. In specific, 0%, 20%, 40% and 60% of PER is selected, meaning that in each case the selected percentage of messages are considered lost. The message loss is determined at the beginning of the transmission so that randomly messages are dropped based on the PER.

The collected results are analysed under the scope of distance and time efficiency of each method. On the one hand, the total distance travelled by the two vehicles is chosen as a metric of energy efficiency. One of the most energy consuming actions of a vehicle is moving, as the motors are demanding in power. Minimising the distance each robot has to travel, means that the energy consumption is minimised. The energy saved can be used to perform other tasks, or perform more extended missions without the need for recharging.

On the other hand, time efficiency is vital regarding mission execution. Sometimes, the missions the robots are trying to solve are time critical, requiring results as fast as possible. In addition to that, lowering the time required to accomplish a mission can reduce the costs and effort required for a given mission. This can be achieved as the users of the robotic systems will spend less time on the field collecting data, allowing them to produce results faster.

The results regarding the distance efficiency of each method for different values of error rates can be seen in figure 5.12. When there is no error in communication the greedy and k-Means based methods are performing almost equally good, while the mTSP based method performed on average about 6% worse. The same

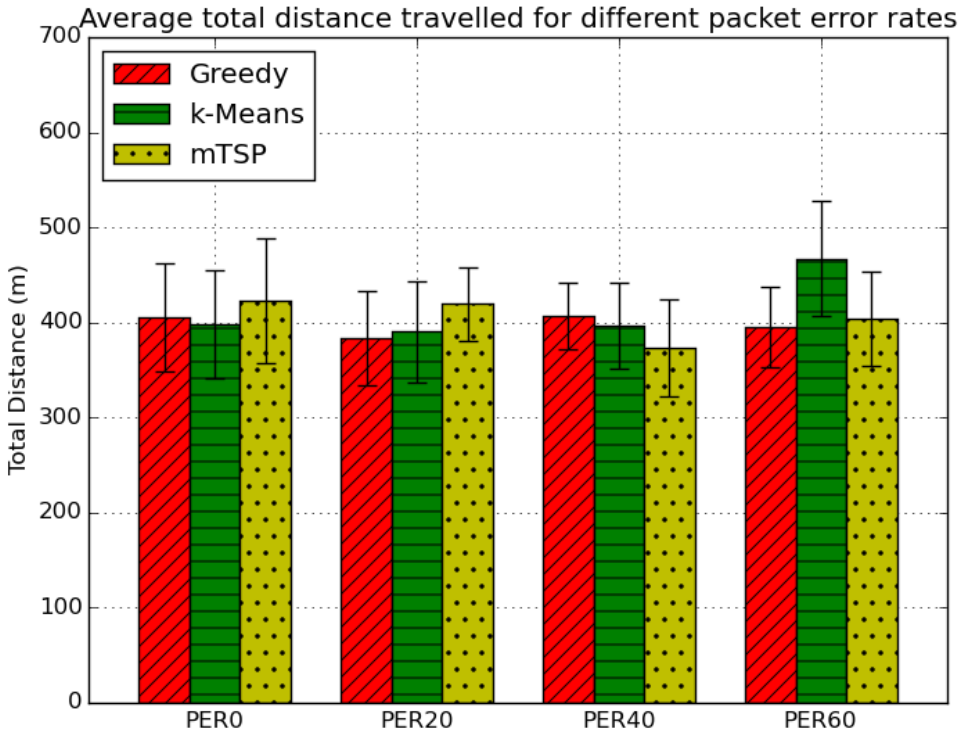


Figure 5.12. Average total distance travelled for different values of packet error rate. Error bars represent one standard deviation. The greedy method performs consistently good. The k-Means method is better in lower packet error rates, while the mTSP performs better in higher packet error rates.

behaviour is observed in low error rates (i.e. 20%). As the error rates grow to 40% and 60%, the mTSP method performs better. It constantly shows better results than the k-Means based method, while it performs at least equally good as the greedy method. In general, the performance of the greedy method regarding the distance is noteworthy.

The time efficiency results for each method and error rate are presented in figure 5.13. In general, one can see that the time to perform the mission increases with the error rate. That is expected, as the information synchronisation is harder. Additionally, it is visible that the k-Means method is consistently performing better than the other methods. The mTSP method is always really close to the k-Means, in the region of less than 5% difference. Finally, the greedy method performs always the worst on average.

Having the above quantitative result sets and having observed the simulated

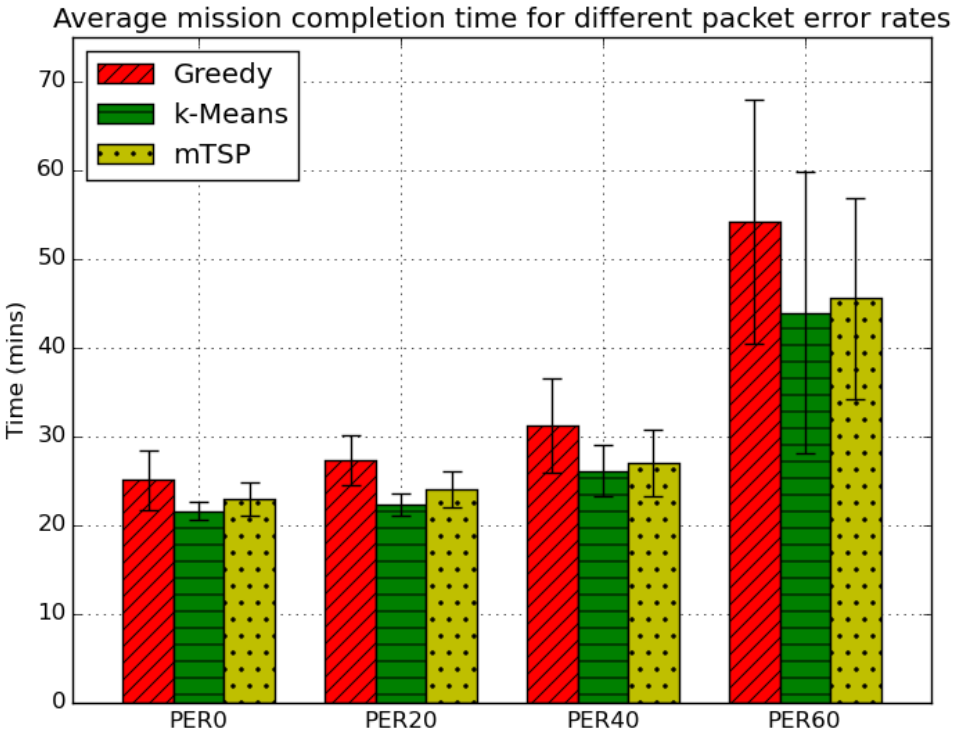


Figure 5.13. Average total mission time for different values of packet error rate. Error bars represent one standard deviation. The k-Means method is constantly giving the best results. The mTSP method is very close to that. The greedy method is the worst performer in general.

vehicle behaviour, one can attempt to perform a qualitative analysis. A general trend is that the greedy method requires less total distance to be travelled by the inspection vehicles, while the total mission time is higher than the other methods. This can be explained by the way the assignment is performed in the greedy allocation. The greedy method is assigning the targets to the closest robot, and then the targets are visited in an optimised way provided by an internal TSP solver. In this way, the number of targets assigned to each robot can be unbalanced, and one robot may remain idle for long periods. There are cases that one robot performed the whole mission alone, while the second remained at the starting position.

Another issue that the greedy algorithm is facing emerges from the inconsistencies in the world model of each robot caused by the latent and erroneous communications. There are cases where the robots had old information regarding

the position of each other and wrongly assigned a target. As the greedy scheme is not allowing for a reallocation, this caused problems with the mission execution, especially in high packet error rates. The solution to that is to set a timer of the expected end of the mission with the current information. If this timer expires and the mission is not finished then a total reallocation is taking place.

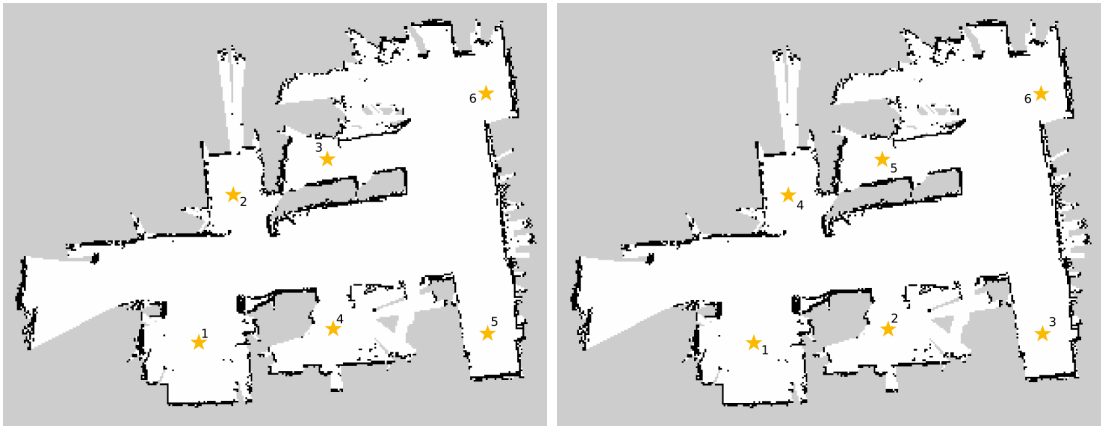
Comparing the k-Means with the mTSP approach one can see that they are almost equally good, with the exception of high communication error rates. In that case, the k-Means highly underperformed in terms of distance travelled.

5.5.2 Real vehicle results

In the previous section, the simulated experiments showed that the two non-greedy methods performed equally well. Unfortunately, the mTSP algorithm for the optimised allocation does not take into consideration the dynamics of the mission as each plan is generated considering the robots are at the starting point. When the plan is devised, and targets are allocated to robots, an internal TSP solver is used to visit them optimally from the current position of the robot. This can lead to suboptimal allocations as the current position of the robot is not considered and eventually may require to travel longer distances.

As already described before, to overcome this issue the use of an MDmTSP instance can be used that will allocate the available tasks taking into account both the current robots position and the final position that they need to be after finishing the mission. This approach is evaluated using real-world experiments are run in the space of the Ocean Systems Laboratory, using real robots as inspection vehicles. Two different virtual target detection patterns are used. The map of the lab, generated by manually driving one robot around and using its sensor, along with the two detection patterns can be seen in figures 5.14a and 5.14b. For this set of experiments, it was assumed that the robots already have a map of the environment.

The target detection simulation is done similarly as in the previous section. Predefined virtual targets are discovered at a regular interval of one minute, and the simulation of the inspection and classification phases is defined by setting a 1.5 minute period where the robot was idle. Regarding the communication delays and errors, the same simulator is used. Its settings are fixed to a packet size of



(a) **(b)**

Figure 5.14. Maps of the Ocean Systems Laboratory created by a Turtlebot using a Kinect sensor. The stars represent the virtual targets the robots have to inspect with the numbers denoting the order of discovery.

512 bytes, while the time to travel from source to destination is set to 0.5 seconds. In addition to that, the timeslot for the TDMA MAC was set to three seconds. Finally, the packet error rate was set to 10%.

For this set of experiments, four different allocation schemes are examined. The first one is a greedy approach as the one used in the previous section. The second is a k-Means based one that clustered the targets taking into account only the Euclidean distance between them. Finally, two different versions of the MDmTSP are tested. The one aims at optimal allocation setting the parameter K that controls the minimum targets allowed to be visited by a vehicle to 1. The second is a fair flavour of the MDmTSP where the minimum targets per robot aim at an equal workload.

The comparison metrics are again the total distance travelled by both robots and the time required to classify all the targets. The distance is calculated by integrating the navigation data provided by the robots. The total time is computed from the beginning of the mission until the time the classification information of the last target is passed to the whole team. It must be noted that the total distance contains the distance required to travel to the ending point of the mission, while this is not added to the total time.

The results for the different allocation methods used on the first detection pattern as shown in 5.14a are presented in table 5.1. This table presents the average total distance and mission completion time over five trials. The best

Table 5.1. Average total distance and time execution for the four different allocation methods under the first detection pattern.

Allocation method	Greedy	k-Means	MDmTSP	
			K=1	K=fair
Average distance (m)	**56.83**	69.64	62.99	68.93
Average time (mins)	18.83	10.81	14.44	**10.71**

method regarding the total distance travelled is the greedy one. It is followed by the unfair MDmTSP and last, are the k-Means and the fair MDmTSP. Regarding the time to mission completion, the things are exactly the opposite. The k-Means and the fair MDmTSP are the fastest to finish. The unfair MDmTSP is third, and the last one is the greedy method.

This behaviour reflects what has been observed in the simulation trials and can be seen in figure 5.15 and in videos of the execution[1]. In the greedy method, only one robot is used as shown in figure 5.15a. This leads to a path optimised execution as the internal TSP solver optimised the distance travelled but the execution time is almost double than the best methods. Using the unfair MDmTSP method, one vehicle has to do more work as it is shown in figure 5.15c, and is reflected in the distance and execution time. The k-Means and fair MDmTSP methods are shown in figures 5.15b and 5.15d. In both cases, the targets are equally distributed between the two inspection vehicles and optimal paths are travelled. This way the execution time is minimised despite the maximal total travelled distance.

Table 5.2. Average total distance and time execution for the four different allocation methods under the second detection pattern.

Allocation method	Greedy	k-Means	MDmTSP	
			K=1	K=fair
Average distance (m)	**47.43**	78.54	68.44	62.65
Average time (mins)	18.93	11.68	12.87	**10.95**

The average results over five trials for the second detection pattern can be seen in table 5.2. As with the previous pattern, the greedy method performs the best in terms of distance efficiency. The second method is the fair version of the MDmTSP. Following is the unfair method of the MDmTSP, and the last method

[1] http://osl.eps.hw.ac.uk/videos/icra16mmtsp

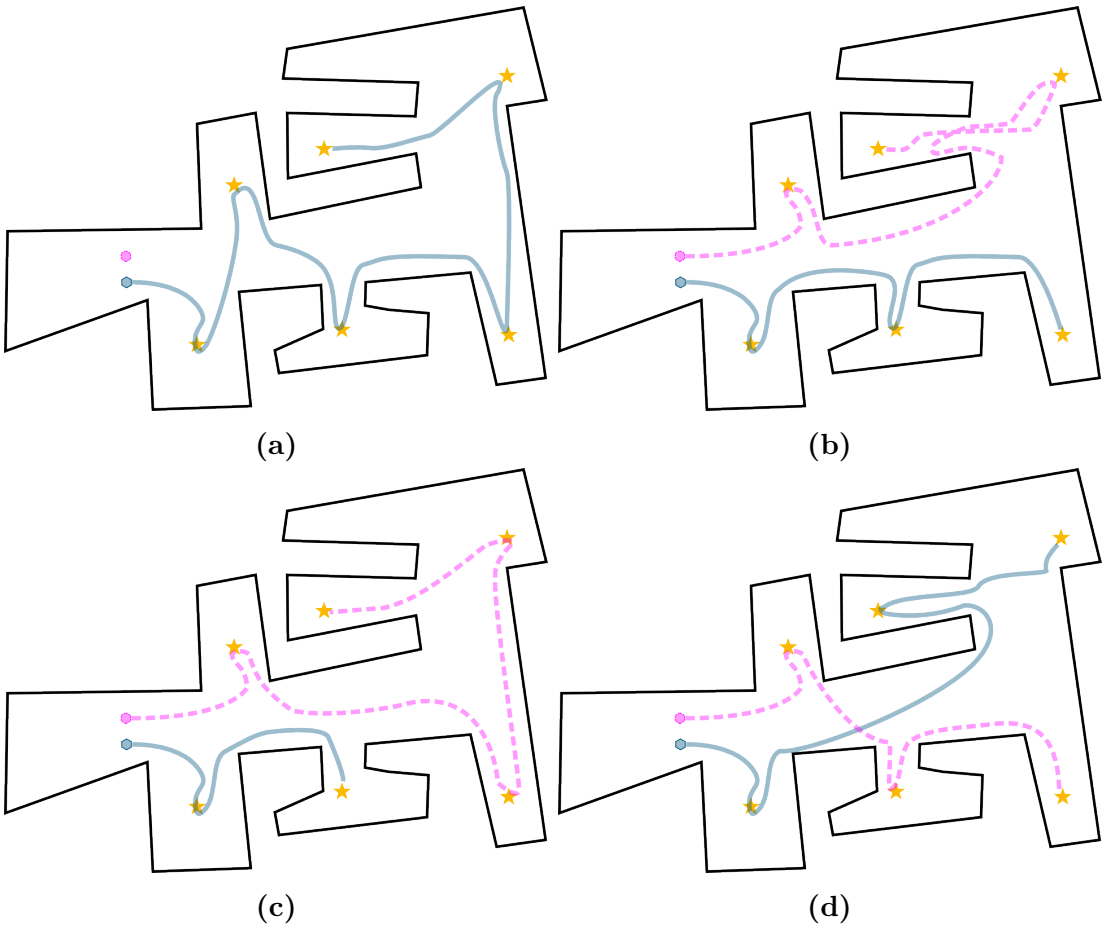


Figure 5.15. Paths traversed by the two robots for the first target detection pattern. (a) shows the greedy task allocation, (b) is the k-Means task allocation, (c) shows the simple MDmTSP task allocation and (d) shows the fair MDmTSP task allocation.

is the k-Means one. In terms of time, the fair MDmTSP is the most efficient, with the k-Means and unfair MDmTSP closely following. The last one is again the greedy method.

Again the greedy approach uses one vehicle as can be seen in figure 5.16a as all the targets to be inspected are always closer to one of the two. The k-Means heuristic having no knowledge of the structure of the environment, groups the targets in a way that requires more distance to be travelled as seen in figure 5.16b. Nevertheless, since both vehicles are used the execution time is much better than the greedy approach. The unfair version of the MDmTSP presented in 5.16c performs close to the k-Means heuristic showing similar allocation. Finally, the fair MDmTSP, shown in figure 5.16d, manages to cluster the targets and provide an efficient solution.

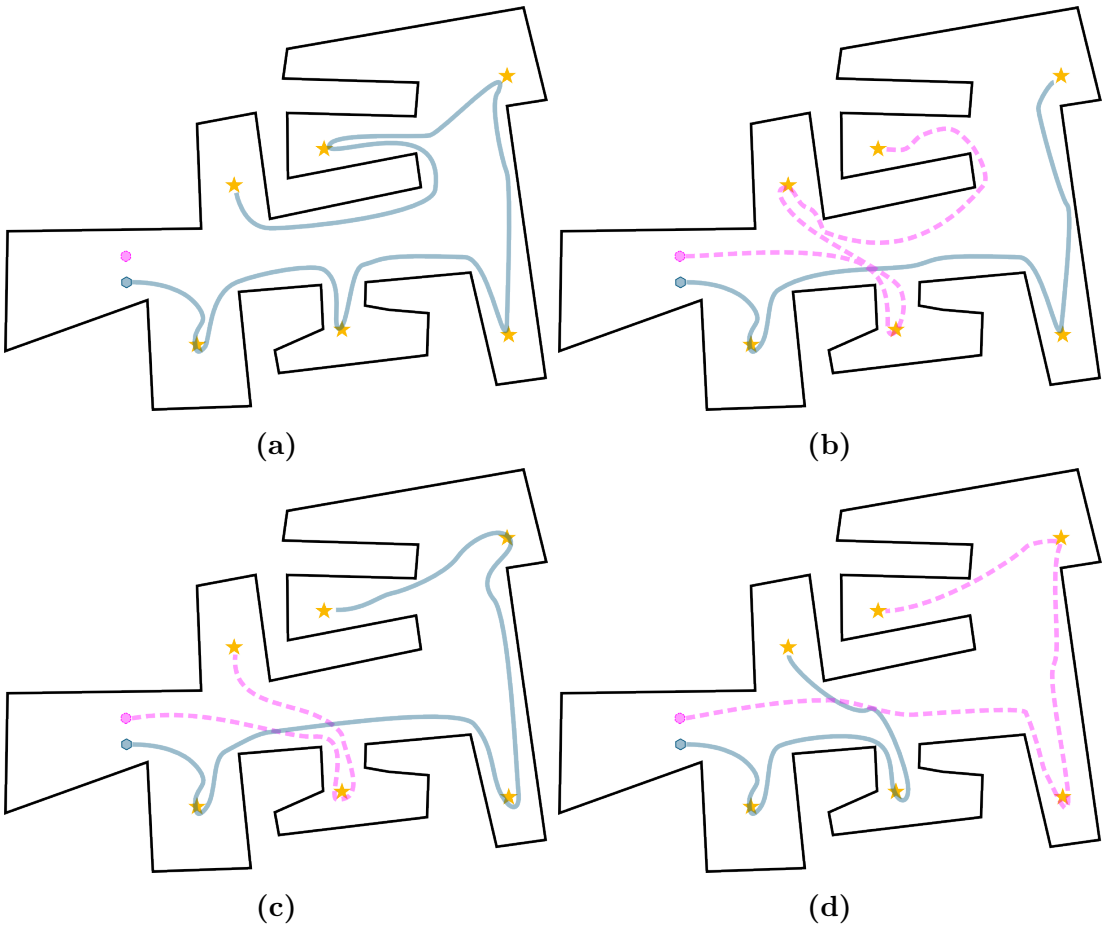


Figure 5.16. Paths traversed by the two robots for the second target detection pattern. (a) shows the greedy task allocation, (b) is the k-Means task allocation, (c) shows the simple MDmTSP task allocation and (d) is the fair MDmTSP task allocation.

In general, it can be seen that the order of the discovered targets can affect the performance of the different allocation methods. Heuristics can still perform close to optimal regarding the execution time but requiring more distance to be travelled. The optimisation based task allocation performs equally or better than the heuristics. In addition, it offers a method to tune its behaviour based on user preferences on execution time or the total travelled distance of the team.

5.6 Summary

This chapter presented the MRTA problem under communication constraints for inspection tasks. Such tasks are mine countermeasures or underwater archaeology missions. A formal analysis of the problem was performed, and methods from

the optimisation literature were presented. An experimental evaluation was conducted both in simulation and with real robots showing the performance of the optimisation based MRTA in various communication error levels. In general, it performs equally or better than simple heuristic methods from the literature.

Chapter 6

Sample collection optimisation

The previous chapter studied an optimisation based MRTA technique for performing inspection missions under communication constraints. Another important type of missions is the one where robots need to collect sampling data over an area. These sensing missions are often used to monitor various phenomena happening in the environment. For example, phenomena like algal blooms, oil spills and chemical pollution need to be monitored and their extent to be estimated [18]. However, operating in the real environment, robots have to cope with its dynamic nature and many constraints it may impose on the mission execution.

This chapter focuses on such missions that require the optimisation of the sampling strategy while respecting some budget constraints. As a budget constraint one can see the maximum time that is available to complete the mission. For example, in emergencies, time is of the essence. It is required to gather the maximum amount of information in a limited amount of time so that a response plan can be devised. Likewise, the maximum amount of energy that is allowed to be spent can be a form of resource constraint. For example, in a highly tidal environment, the currents developed are strong and affect the mission execution energy and time [4]. In such cases, it is required to maximise the sensing outcome with a limited amount of energy, as well as, to be able to replan a maximising trajectory online. Online replanning is also required in longer scale missions when an area needs to be monitored periodically. The behaviour of the vehicle needs to be adapted to the current environmental conditions of each measurement cycle. In the missions studied in this work, one or more AUVs will take part and try to optimise their behaviour. It should be noted that this work focuses on underwater vehicles, but

the proposed approaches can be easily transferred to any domain.

At the beginning a *Mixed Integer Quadratic Problem* (MIQP) formulation for the sampling optimisation problem described in chapter 2 will be detailed. Next, *Genetic Algorithm* (GA) based heuristics are presented for solving the optimisation problem in an online manner for one or more sampling vehicles. Finally, the proposed heuristics are evaluated in benchmark sampling scenarios against the MIQP approach.

6.1 Mixed Integer Quadratic Programming Formulation

In section 2.2 the problem of estimating a scalar field was described. In that problem, there is a set of sampling points that need to be visited while respecting some cost constraint. One can represent the problem using a fully connected undirected graph where each vertex is a sampling point. The weights of the edges between the vertices represent the cost of going from one vertex to the other. The optimisation problem then is to find a subset of these vertices should be visited that maximises the gained reward while respecting the cost.

As it was mentioned, the Gaussian process probabilistic method for field estimation assumes that there is a correlation between the measurements. This correlation information can be taken into account in the optimisation process providing better results. A method that does that is presented in [10], [11] where a variant of the orienteering problem is used for persistent monitoring tasks using Unmanned Aerial Vehicles. The presented approach introduces the *Correlated Orienteering Problem* (COP), a problem that takes into account the correlation of information among the sampling points, while respecting some budget constraints. In that work, a MIQP formulation is provided. A modified version of it where the vehicle must start and finish in user-defined points is subsequently presented.

$$\max \sum_{i \in V} (r_i x_i + \sum_{v_j \in N_i} r_j w_{ij} x_i (x_i - x_j)) \tag{6.1}$$

$$s.t. \sum_{i \in V \setminus \{s\}} x_{is} = \sum_{i \in V \setminus \{f\}} x_{fi} = 0 \tag{6.2}$$

$$\sum_{i \in V \setminus \{s\}} x_{si} = \sum_{i \in V \setminus \{f\}} x_{if} = x_s = x_f = 1 \tag{6.3}$$

$$\sum_{i \in V \setminus \{sf\}} x_{ik} = x_i \leq 1 \qquad \forall k \in V \setminus s, f \tag{6.4}$$

$$\sum_{i \in V \setminus \{sf\}} x_{ki} = x_i \leq 1 \qquad \forall k \in V \setminus s, f \tag{6.5}$$

$$\sum_{i \in V \setminus f} x_{ik} = \sum_{i \in V \setminus s} x_{ki} \qquad \forall k \in V \setminus s, f \tag{6.6}$$

$$\sum_{i \in V} x_i c_i + \sum_{j \in V} c_{ij} x_{ij} \leq C_{max} \tag{6.7}$$

$$u_i - uj + 1 \leq (|V| - 1)(1 - x_{ij}) \qquad \forall i, j \in V, i \neq j \tag{6.8}$$

$$0 \leq u_i \leq |V| \qquad \forall i \in V \tag{6.9}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V \tag{6.10}$$

The objective function, seen in (6.1) is the sum over the rewards of all the visited vertices. The set V includes all the vertices. The reward for visiting vertex i is described by variable r_i, while x_i is a binary variable denoting that a vertex i is visited in a solution. As mentioned before, the COP considers the cases where there is a correlation between the vertices. This is translated in profits even from unvisited vertices based on some function. These vertices form the neighbourhood of vertex i, represented by N_i. The reward based on this correlation is added to the utility obtained by visiting vertex i. It is calculated as the sum of the rewards for visiting each vertex j in the neighbourhood, multiplied by a weight w_{ij} depending on vertices i and j and the quadratic term $x_i(x_i - x_j)$. This quadratic term ensures that extra reward, produced by correlation, is only added for vertices that are not visited in the current solution.

In the version of the COP presented in this work it is assumed that the agent must start and finish in user defined points. This is enforced by constraints (6.2) and (6.3). Here the binary variable x_{si} denotes that vertex i is visited right after the starting vertex s and x_{if} shows that the finishing vertex is after vertex i. In general x_{ij} is a binary variable denoting that a path exists from vertex i to vertex j.

For all the other vertices constraints (6.4), (6.5) and (6.6) are applied. Constraints (6.4) and (6.5) allow vertices to be left out from a solution and enforce that a vertex is visited at most once. Constraint (6.6) ensures that if a vertex is

visited it must continue the path to another vertex, unless it is the user defined finish vertex.

Constraint (6.7) bounds the vehicle to use a maximum amount of budget. In constraint (6.7), the term $x_i c_i$ represents a fixed cost of visiting vertex i. The cost of travelling from vertex i to the next vertex is added to that, as denoted by $c_{ij}x_{ij}$. It should be noted that the fixed cost of visiting a vertex is not added for the user-defined start and finish vertices.

Constraint (6.8) allows only a single path to be generated, preventing smaller disjoint tours. Finally, constraints (6.9) and (6.10) bound the specific problem variables to a certain allowed range.

A sample solution for the COP can be seen in figure 6.1. It can be seen that the path is spread over all the area trying to maximise the points reached by correlation. The reward gathered by correlation from each vertex is calculated using a weight shown in (2.8). For this example the value of l was set to two; this results in a reward of 20.7.

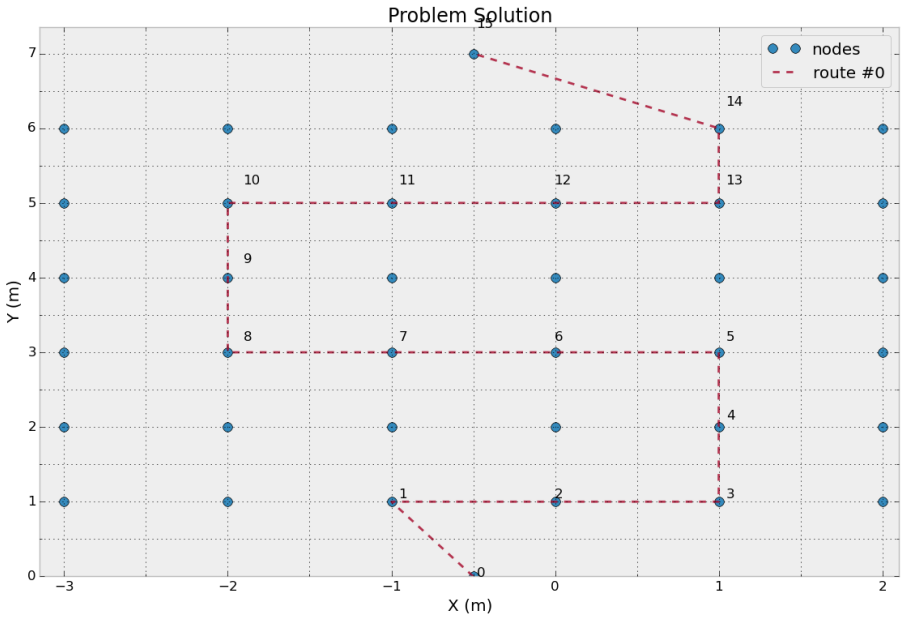


Figure 6.1. Solution to a COP instance for a 6x6 grid. Starting point was set to be $(-0.5, 0)$ and ending point was $(-0.5, 7)$. The maximum allowed cost was 30 units. The reward gathered was 20.7.

When multiple vehicles are available, the COP can be extended to provide paths that maximise the team performance. The problem is then named the

Correlated Team Orienteering Problem (CTOP). A MIQP formulation for that has also been presented in the work of [10], [11]. The formulation maximises the following objective function:

$$\sum_{i \in V} (r_i x_i + \sum_{v_j \in N_i} r_j w_{ij} x_i (x_i - x_j)) \tag{6.11}$$

For any given solution the objective function represents the sum of the reward of each of the visited vertices in that solution. All the vertices are included in set V. The variable r_i represents the reward for visiting vertex i in a solution. To encode that a vertex i is visited in a given solution, the binary variable x_i is used. In addition to the reward received for visiting a vertex i, an additional reward is obtained from its neighbourhood N_i. This reward is calculated by summing the reward of each unvisited neighbour j multiplied by a weight w_{ij}. To ensure that only reward from unvisited vertices will be added, the quadratic term $x_i(x_i - x_j)$ is used.

To enforce that m vehicles start and finish in user defined points, constraints (6.12) and (6.13) are used.

$$\sum_{i \in V \setminus \{s\}} x_{isk} = \sum_{i \in V \setminus \{f\}} x_{fik} = 0, \qquad \forall 1 \leq k \leq m \tag{6.12}$$

$$\sum_{i \in V \setminus \{s\}} x_{sik} = \sum_{i \in V \setminus \{f\}} x_{ifk} = m, \qquad \forall 1 \leq k \leq m \tag{6.13}$$

In this formulation a binary variable x_{ijk} is showing that a path exists from vertex i to vertex j in the path of vehicle k. Specifically, x_{sik} shows that any vertex i is visited right after the starting vertex s for the path of vehicle k. Likewise, variable x_{ifk} defines that the finishing vertex is after vertex i. For all the other vertices in the solution constraints (6.14)-(6.18) are applied.

$$\sum_{j \in V \setminus \{sf\}} x_{ijk} \leq 1, \qquad \forall i \in V \setminus s, f, 1 \leq k \leq m \tag{6.14}$$

$$\sum_{j \in V \setminus \{sf\}} x_{jik} \leq 1, \qquad \forall i \in V \setminus s, f, 1 \leq k \leq m \tag{6.15}$$

In the solution, each vertex is visited at most once, as it is ensured from constraints (6.14) and (6.15). This behaviour is enforced by allowing at most one path entering and one leaving in each of the vertices in V, other than the start and finish. This happens for all the m paths.

$$\sum_{k=1}^{m} \sum_{j \in V \setminus \{sf\}} x_{ijk} = x_i \leq 1, \qquad \forall i \in V \setminus f \tag{6.16}$$

$$\sum_{k=1}^{m} \sum_{j \in V \setminus \{sf\}} x_{jik} = x_i \leq 1, \qquad \forall i \in V \setminus s \tag{6.17}$$

$$\sum_{j \in V \setminus \{sf\}} x_{ijk} = \sum_{j \in V \setminus \{sf\}} x_{jik}, \qquad \forall i \in V, 1 \leq k \leq m \tag{6.18}$$

Constraints (6.16) and (6.17) make sure that if a vertex j is visited in a solution, a path that connects j with a next vertex exists. Constraint (6.18) prevents a vertex having a path from one vehicle entering and the path of a different vehicle exiting it. The maximum amount of resources used is enforced by constraint (6.19).

$$\sum_{i \in V} \sum_{j \in V} (c_{ij} + c_i) x_{ijk} \leq C_k, \qquad \forall 1 \leq k \leq m \tag{6.19}$$

Term c_i represents a fixed cost of performing sensing in vertex i. To that, the cost of travelling from vertex i to the next vertex c_{ij} is added. Then the cost is multiplied by the binary variable x_{ijk}, used to denote the existence of a path between i and j in the path of robot k. The sum of all these costs must be less or equal to the maximum allowed resource usage for robot k. It must be noted that the fixed cost of operations of the user-defined start and finish vertices is zero.

$$\begin{aligned} &u_{ik} - u_{jk} + 1 \leq (|V| - 1)(1 - x_{ijk}) \\ &\forall i, j \in V, i \neq j, \forall 1 \leq k \leq m \end{aligned} \tag{6.20}$$

$$\begin{aligned} &0 \leq u_{ik} \leq |V| \\ &\forall i \in V, \forall 1 \leq k \leq m \end{aligned} \tag{6.21}$$

Finally, constraints (6.20) and (6.21) are subtour elimination constraints, al-

lowing only a single tour to be generated for each robot.

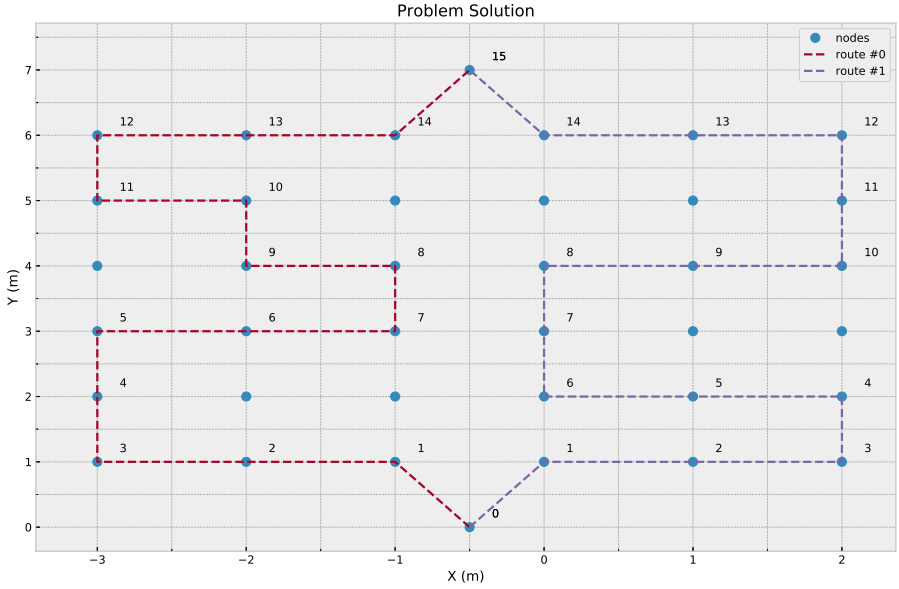


Figure 6.2. Solution to a CTOP instance for a 6x6 grid. Starting point was set to be $(-0.5, 0)$ and ending point was $(-0.5, 7)$. The maximum allowed cost was 30 units per agent, while there were two agents available. The reward gathered was 20.7.

In figure 6.2 is presented. The solution was obtained using the aforementioned MIQP formulation. The two agents started from $(-0.5, 0)$ and ended at $(-0.5, 7)$. Each agent had a budget of 30, and the optimal solution gave a profit of 31.279.

6.2 Genetic Algorithm Heuristic

In the work presented by [10], [11] the COP is solved using the commercial solver Gurobi [13]. There, the anytime property of that solution is stressed, meaning that the execution can stop anytime and get a solution with some optimality guarantee. In [109] the method was adapted to include constraints for the starting and finishing the missions on specific vertices. There it was observed that having a close to the optimal solution is prohibitive for online use as its time complexity is high even for moderate instances.

To overcome the problem of the time complexity, the work presented in [110] introduced a GA based heuristic. The proposed algorithm is the first presented

heuristic for the COP. Therefore, most of the parts are newly presented. In particular, unlike other methods, it heavily uses 2-opt optimisation [111] in every solution step. This way it tries to increase the number of sampling points in the solution. Additionally, it presents a novel mutation method. In previous works using GA for solving the standard orienteering problem, a mutation was either adding or removing vertices (genes) from a solution. This work instead proposes that a vertex may be beneficial to be swapped with one of its neighbouring vertices. This technique is used whenever a chromosome is very close to its maximum length, and new vertices cannot be inserted without violating the cost constraints. Moreover, to have better performance, the mutations are performed in parallel, taking advantage of modern multicore architectures. Finally, it is proposed to try further to optimise the fittest chromosome after the genetic algorithm finishes.

Algorithm 6.1 Genetic Algorithm for the Correlated Orienteering Problem

Input: $TravelCosts, Rewards, MaxCost, Start, Finish$
Output: $FittestChromosome$
1: InitialisePopulation
2: **while** $generation \leq MaxGeneration$ **do**
3: SelectNewPopulation
4: Crossover
5: Mutate
6: Select and Optimise $FittestChromosome$
7: **return** $FittestChromosome$

As it can be seen from algorithm 6.1 the optimisation process consists of five steps. Initially, a population of chromosomes is generated. Each chromosome represents a candidate solution. It is encoded as a vector containing the indices of the vertices to be visited in that solution.

In the second step, the chromosomes are going through repetitive operations until a maximum number of generations is reached. Each generation starts by selecting a new population based on a natural selection process applied to the previous generation. Then, a number of chromosomes are allowed to reproduce and generate offsprings. Finally, a number of randomly selected individuals are going through a mutation process. When the maximum number of generations is reached, the fittest chromosome is selected. Then some final optimisation is performed on that chromosome, and it is returned as the solution to the problem. The fitness of each chromosome is calculated as shown in (6.22), which is proposed

in [70].

$$Fitness = Reward^3/cost \tag{6.22}$$

Algorithm 6.2 Initialise Population

Input: $PopSize, TravelCosts, MaxCost, Start, Finish$
Output: $Population$

```
1:  Population ← [∅]
2:  for i ← 1, PopSize do
3:      CR ← [Start]
4:      V ← [1, ..., NumVertices]
5:      while !Done && V ≠ [∅] do
6:          v ← SelectRandom(V)
7:          if cost_{CR+v} ≤ MaxCost then
8:              CR ← CR + v
9:              V ← V \ v
10:         else
11:             Done ← True
12:     CR ← CR + Finish
13:     CR ← TwoOpt(CR)
14:     Population ← Population + CR
15: return Population
```

Algorithm 6.2 describes the first step in the GA heuristic method, namely, the population initialisation. The population is initialised as an empty list. It is then populated with a user-defined $PopSize$ chromosomes. Each chromosome is generated in the following way. Initially, the chromosome contains only the starting vertex. Next, a random vertex v is selected from the set V of free vertices. If the current cost of the chromosome plus the cost of travelling to v is lower than the maximum allowed cost $MaxCost$, the vertex is appended to the chromosome. The process is repeated until the $MaxCost$ is reached or there are no more free vertices left. After that, the finish vertex is appended to the chromosome. The next step is to perform a *2-opt* optimisation to the chromosome and append it to the population list. This process is repeated $PopSize$ times.

After the population initialisation, the main iterative part of the algorithm begins. The first step is the generation of a new population based on a natural se-

Algorithm 6.3 Select new population

Input: $Population, TourSize$
Output: $NewPopulation$

```
1: Pop ← Population
2: NewPopulation ← [∅]
3: for i ← 1, PopSize do
4:     Candidates ← SelectRandom(Pop, TourSize)
5:     Fittest ← GetFittest(Candidates)
6:     NewPopulation ← NewPopulation + Fittest
7: return NewPopulation
```

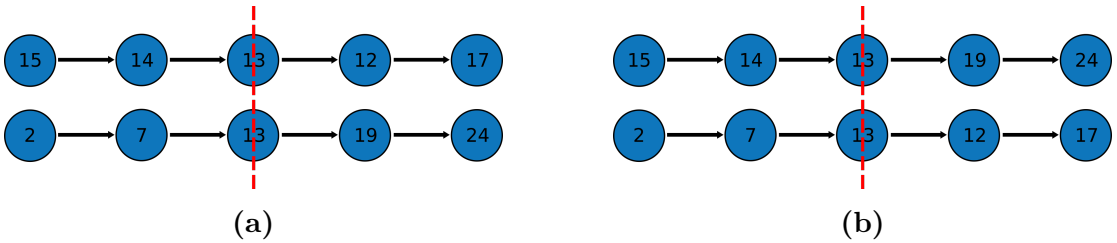


Figure 6.3. Two chromosomes before and after the crossover operation. The two parents have a common gene. The operation takes place by swapping the paths after that gene.

lection process. This is performed using a standard technique in the GA literature called *Tournament selection* and presented in algorithm 6.3. This algorithm starts with an empty new population. It is then populated by performing $PopSize$ tournaments aiming to select candidates based on their fitness. For each tournament $TourSize$ candidates are randomly selected from the old population. The fittest of the candidates is then winning and is appended to the new population.

The next step, following the population selection, is the crossover operation allowing parts of the population to reproduce and generate offsprings. This process is described in algorithm 6.4. To perform the crossover operation, initially, two individuals are selected randomly from the population. Then the set of common genes of the two parents is calculated. If the set is empty the combination is incompatible for crossover and the operation is not performed. If the set is not empty, then a random gene is selected. The two parent chromosomes are then sliced at that gene, and two offsprings are generated by swapping the parent's tail parts. The operation can be seen in figure 6.3.

The offsprings are then going through *2-opt* optimisation, any duplicate genes

Algorithm 6.4 Population crossover

Input: $Population, NumCrossovers$
Output: $Population$

$Pop \leftarrow Population$
for $i \leftarrow 1, NumCrossovers$ **do**
    $CR^1, CR^2 \leftarrow \text{SelectRandom}(Pop, 2)$
    $I \leftarrow \text{GetCommonGenes}(CR^1, CR^2)$
    if $I \neq [\varnothing]$ **then**
        $RG \leftarrow \text{SelectRandom}(I, 1)$
        $OF^1 \leftarrow [CR^1_{[1,RG]}, CR^2_{[RG,end]}]$
        $OF^2 \leftarrow [CR^2_{[1,RG]}, CR^1_{[RG,end]}]$
        $OF^1 \leftarrow \text{TwoOpt}(OF^1)$
        $OF^2 \leftarrow \text{TwoOpt}(OF^2)$
        $OF^1 \leftarrow \text{RemoveDuplicates}(OF^1)$
        $OF^2 \leftarrow \text{RemoveDuplicates}(OF^2)$
        $Feas^1 \leftarrow \text{CheckFeasibility}(OF^1)$
        $Feas^2 \leftarrow \text{CheckFeasibility}(OF^2)$
        if $Feas^1$ && $Feas^2$ **then**
            $Pop_{CR^1} \leftarrow OF^1$
            $Pop_{CR^2} \leftarrow OF^2$
        else
            if $Feas^1$ **then**
                $Pop_{CR^2} \leftarrow \text{GetFittest}(CR^1, CR^2)$
                $Pop_{CR^1} \leftarrow OF^1$
            else if $Feas^2$ **then**
                $Pop_{CR^1} \leftarrow \text{GetFittest}(CR^1, CR^2)$
                $Pop_{CR^2} \leftarrow OF^2$
return Pop

are removed, and their feasibility is calculated by comparing them to the $MaxCost$. If both are feasible, then their parents are replaced by them in the population. If only one is feasible, it replaces the respective parent, and the other parent is replaced by the fittest of the two parents. Finally, if the offsprings are not feasible, then the parents are remaining in the population. This process is repeated $NumCrossovers$ times.

Algorithm 6.5 Population mutation

Input: $Population, PopMut, CrMut, AddProb$
Output: $Population$

$Pop \leftarrow Population$
$CR \leftarrow \text{SelectRandom}(Pop, PopMut)$
for $i \leftarrow 1, PopMut$ **do**
    $M \leftarrow \text{TwoOpt}(CR_i)$
    $M \leftarrow \text{RemoveDuplicates}(M)$
    $V \leftarrow [1, \ldots, NumVertices]$
    $FV \leftarrow V \setminus M$
    for $j \leftarrow 1, CrMut$ **do**
        $p \leftarrow \text{GetRandom}(0, 1)$
        if $p \leq AddProb$ **then**
            if $Cost_M \geq 0.99 * max\_cost$ **then**
                $v \leftarrow \text{SelectRandom}(M, 1)$
                $n \leftarrow \text{GetMaxFreeNeighbour}(M_v)$
                if $Fit_{M_v} \leq Fit_{M_n}$ **then**
                    $M_v \leftarrow M_n$
            else
                if $FV \neq [\varnothing]$ **then**
                    $v \leftarrow \text{SelectRandom}(FV, 1)$
                    $idx \leftarrow \text{FindBestInsertion}(v, M)$
                    $M \leftarrow M_{[} \ldots, idx, v, idx + 1, \ldots]$
        else
            if $cost_M \geq 0.99 * max\_cost$ **then**
                $v \leftarrow \text{GetMinLossGene(M)}$
                $M \leftarrow M_{[} \ldots, v - 1, v + 1, \ldots]$
    $Pop_{CR_i} \leftarrow M$
return Pop

The final step in the iterative part of the GA optimisation process is the mutation step. It is described in algorithm 6.5. Initially $PopMut$ chromosomes are randomly selected from the population. Then each chromosome goes through the

following mutation process. The first step applies a *2-opt* and removes any duplicate genes. Then, the set of free vertices is calculated. In each mutation process, $CrMut$ operations take place.

The mutation operations can be either the addition or the removal of a gene with some probability. The addition operation has two behaviours depending on the cost of the chromosome. The first behaviour occurs when the chromosome has a cost close to the $MaxCost$. In that case, a random gene is selected from the chromosome. A search over the free neighbours of that chromosome is performed, and the fitness of replacing the selected gene with its neighbour is calculated. If that fitness is higher than the current fitness, the mutation happens. The second behaviour, happening in chromosomes with low cost, selects a random gene from the free genes. Then it finds the best position for insertion and inserts it there. The removal operation happens only in chromosomes with a cost close to the $MaxCost$. In those cases, the gene whose removal causes the minimum fitness loss is selected. It is then removed from the chromosome.

After the maximum number of generations is reached the fittest chromosome is selected. On that chromosome, a final optimisation round is applied. In this last round, each gene is traversed and is checked if replacing it with any of its neighbours results in higher fitness feasible chromosome. If yes a swap is performed. After this final step, the chromosome is the solution to the COP instance.

When the sampling is performed by more than one vehicles, the problem is described as the CTOP according to [10], [11]. Following is a GA based method to solve the CTOP online, as solving the previously presented MIQP with any commercial solver is prohibitively expensive.

Algorithm 6.6 Genetic Algorithm for the Correlated Team Orienteering Problem

Input: $TravelCosts, Rewards, MaxCosts, Start, Finish$
Output: $FittestChromosome$

```
1: InitialisePopulation
2: while gen ≤ MaxGen or FittestChromosome stable for 10 generations do
3:     SelectNewPopulation
4:     Crossover
5:     Mutate
6: select FittestChromosome
7: return FittestChromosome
```

The genetic algorithm presented here uses the chromosome encoding as pre-

sented in [112]. It is composed of a set vectors, or genes, each one of which is representing a path for a single robot. In each vector, the indices of the vertices to be visited by that robot is stored. The whole chromosome, therefore, represents a candidate CTOP solution. An outline of the procedure used during the optimisation can be seen in algorithm 6.6. It can be seen that the optimisation procedure is separated into two distinct phases. The first phase is responsible for the population generation, while the second phase repeatedly applies the operations of selection, crossover and mutation to optimise the population.

In the scope of this work, two different chromosome generation methods are presented. The difference is based on the way the genes, composing the chromosome, are generated. The first method is a random method, similar to the one presented in [110]. It can be seen in algorithm 6.7.

Algorithm 6.7 Random Gene Generation

Input: $TravelCosts, Rewards, MaxCost, Start, Finish, AvailVertices$
Output: $Gene, AvailVertices$

```
1: INSERT(Gene, Start)
2: Done ← False
3: while !Done and AvailVertices ≠ ∅ do
4:     v ← SELECTRANDOM(AvailVertices)
5:     if COSTINSERT(Gene, v) ≤ MaxCost then
6:         INSERT(Gene, v)
7:         AvailVertices ← AvailVertices \ v
8:     else
9:         Done ← True
10: INSERT(Gene, Finish)
11: return Gene, AvailVertices
```

In the random generation method, genes are created by randomly inserting vertices until the maximum cost is reached or until there are no more free vertices to insert. To generate a chromosome this process is repeated a number of times, equal to the number of robots. In the chromosome generation process, the set of available vertices is shared among the genes, so that if a vertex is picked up by a robot, it cannot be assigned to any other robot.

The second gene generation method is named *Nearest Neighbour Randomised Adaptive Search Procedure* (NN-RASP). It is inspired by the *Greedy Randomised Adaptive Search Procedure* (GRASP) that is used in [82] to provide a heuristic

solution to the TOP. This procedure is taking into consideration the peculiarities of the CTOP problem, regarding the correlation of information. It and can be seen in algorithm 6.8.

Algorithm 6.8 Nearest Neighbour Randomised Adaptive Search Procedure

Input: $TravelCosts, Rewards, MaxCost, Start, Finish$
Output: $Gene$

```
1: INSERT(Gene, Start)
2: v ← GETLAST(Gene)
3: N ← GETNEIGHBOURS(v)
4: while GETSIZE(N) > 0 do
5:     for each n ∈ N do
6:         GETDISTANCEWEIGHT(n)
7:         GETNEIGHBOURWEIGHT(n)
8:         GETCOMBINEDWEIGHT(n)
9:     nv ← GETWEIGHTEDRANDOM(N)
10:    if COSTINSERT(Gene, nv) <= MaxCost then
11:        INSERT(Gene, nv)
12:        v ← GETLAST(Gene)
13:        N ← GETNEIGHBOURS(v)
14:    else
15:        N ← {∅}
16: INSERT(Gene, Finish)
17: return Gene
```

This method is based on progressively constructing a path by choosing one of the neighbours of the last inserted vertex in the path. The neighbour to be inserted is chosen based on a categorical distribution where each one has a probability of being chosen. This probability is based on the distance of the neighbouring vertex from all other visited vertices in the solution and the number of free neighbours it has. The distance from other vertices is chosen as this can reduce the overlapping of paths. The number of free neighbours is used as it gives a better reward from correlated information, as well as, it will allow the path to have more options to continue its construction. When a vertex is chosen, it is attempted to be inserted in the gene. If the gene that is resulting is feasible, the vertex is inserted, and the search continues. Else the search stops and the gene is returned. The chromosome generation is again performed by iteratively calling the gene generation methods until paths are generated for all the robots.

To create the population, both chromosome generation procedures are repeated until the specified amount of chromosomes is generated. Following their generation, the chromosomes are evaluated according to algorithm 6.9.

Algorithm 6.9 Chromosome Evaluation

Input: $TravelCosts, Rewards, MaxCost$
Output: $Fitness$
1: **for each** $g \in Genes$ **do**
2: TwoOpt(g)
3: RemoveDuplicates(g)
4: $GeneFit \leftarrow$ GetReward(g)^3/GetCost(g)
5: $Fitness \leftarrow Fitness + GeneFit$
6: **return** $Fitness$

The fitness of each chromosome is defined as the sum of fitnesses of each gene composing it. Each gene goes through a *2-Opt* operation to minimise its cost. This operation is a simple local search method that is based on swapping segments of a path in hope of reducing the path's cost. It is iteratively applied to all segment combinations and stops if no improvement can be achieved. It was first presented in [111] as a method to solve the travelling salesman problem. Then any duplicate vertices that have been already visited by other genes are removed. Its fitness is calculated as the total reward to the third power divided by the gene cost as suggested in [70] for the single agent orienteering problem.

After the population generation is completed, a repetitive process is applied. This process is applied until either the best chromosome is stable for ten generations or a maximum amount of generations is reached. This strategy is chosen because if the chromosome is stable for long enough, it is either on a local maximum that is hard to escape or the global maximum has been reached. Either way, given that the algorithm is used to be applied online, it is useful to stop early and not waste resources.

The aforementioned repetitive process involves selecting a new population based on the previous one. A percentage of the old population is passed to the next generation without selection. This process is described in the literature as *Elitism* and is known to speed up the performance and prevent loss of good solutions during the search process [113]. The rest of the population is selected using *Tournament selection*. This method starts with an empty population, then repet-

itively chooses n random individuals from the previous population and adds the fittest to the new population, until the maximum amount of individuals is chosen.

Following the new population selection, a crossover operation is performed to a percentage of the population. The operation is presented in 6.10 and is a modified version of the one presented in [112]. The crossover operation aims to diversify the population and explore more of the search space.

Algorithm 6.10 Chromosome Crossover

Input: $Parent_1, Parent_2$
Output: $Child_1, Child_2$

```
1: SortGenes(Parent_1)
2: SortGenes(Parent_2)
3: for each i ∈ 1, 2 do
4:     tP_1 ← Parent_1
5:     tP_2 ← Parent_2
6:     while GetNumGenes(Child_i) < numRobots do
7:         p ← SelectRandomParent(tP_1, tP_2)
8:         InsertBestGene(Child_i, p)
9:         RemoveVertices(tP_1, tP_2)
10:        EvaluateAndSortGenes(tP_1, tP_2)
11:    EvaluateChromosome(Child_i)
12: return Child_1, Child_2
```

For this operation to take place, two chromosomes are randomly chosen as the parents of the crossover, and they are replaced by the two produced offsprings. Initially, the genes of each parent are sorted by fitness. Then, the two children are constructed iteratively. The first step of the iteration involves copies of the parents to be created. Next, as long as the number of genes in each child is less than the number of robots, the following process takes place. A parent is selected randomly, and its best gene is removed and inserted to the child. Then, the vertices of this gene are removed from any other genes of the temporary parents. After that, the genes of the parents are re-evaluated and sorted based on their fitness. When the maximum amount of genes is inserted in the child, it is accordingly evaluated. After both offsprings are generated, they are returned in the place of their parents.

Continuing the population crossover, a percentage of the population is evolving by mutating. This aims at attempting to improve the fitness of each gene that composes the mutating chromosome and thus improve the fitness of the chromo-

Algorithm 6.11 Chromosome Mutation

Input: $Chromosome, MaxCost, NumMut, AddProb$
Output: $Chromosome$

```
1: Genes ← GetGenes(Chromosome)
2: for each g ∈ Genes do
3:     for i ← 1, NumMut do
4:         p ← GetRandom(0, 1)
5:         if p ≤ AddProb then
6:             if GetCost(g) ≥ 0.95 * MaxCost then
7:                 v ← SelectRandom(g, 1)
8:                 n ← GetMaxFreeNeighbour(v)
9:                 gn ← Replace(g, v, n)
10:                if Fitness(g) ≤ Fitness(gn) then
11:                    g ← gn
12:            else
13:                if FV ≠ [∅] then
14:                    v ← SelectRandom(FV, 1)
15:                    idx ← FindBestInsertion(g, v)
16:                    Insert(g, v, idx)
17:        else
18:            v ← GetMinLossVertex(g)
19:            g ← Remove(g, v)
20: EvaluateChromosome(Chromosome)
21: return Chromosome
```

some itself. The mutation process resembles the one presented in [110] and can be seen in algorithm 6.11.

For each gene in the chromosome, a total number of $NumMut$ mutations will take place. In the case studied in this work $NumMut$ was set to 10. A mutation can happen in two ways, by either adding a vertex in the gene or by removing one. This is governed by the $AddProb$ probability and was set to be 0.9.

In the first case, there are two behaviours depending on the cost of the gene that is mutated. If the cost is 95% of the maximum cost, an improvement of the solution is attempted by swapping a vertex of the solution with one of its free neighbours. A random vertex in the solution is chosen and is iteratively swapped with all its free neighbours. The swap with the maximum improvement in the gene's fitness is kept as the performed mutation. This is done as it can be hard to insert a new vertex in the solution without violating the cost constraints. In such

a case it is preferable to try and improve the solution using a local search method. In the second behaviour, a free vertex is randomly chosen. Then the best insertion place is found in the path. It is consequently inserted if it does not violate the maximum cost constraint. The best insertion is found using the heuristic value described in the GRASP method in [82].

In the second case, where a removal is performed, the minimum loss vertex for this gene's fitness is found and removed. Likewise, with the insertion, the minimum loss vertex is found using the same heuristic value. After all the $NumMut$ mutations happen for all the genes, the chromosome is evaluated using the method already described and is returned.

After the number of generations reaches the maximum number or if the best chromosome is stable for more than ten generations the algorithm stops and returns the best chromosome found as a solution to the CTOP problem.

6.3 Experimental setup

As described before a sample collection mission requires one or more vehicles to visit predefined sampling points over an area and collect samples. To evaluate the performance of the proposed methods simulations are performed using areas of various sizes. Each region consists of sampling points arranged in a grid. In addition, there are specific starting and ending points. Moreover, each vehicle is given a limited amount of energy to complete its tasks. The overall vehicles' goal is to maximise the gained utility before reaching the energy threshold. To solve the MIQP formulation the Gurobi 6.5 commercial solver [13] is used. The GA heuristic method is implemented in C++11 and compiled using g++ version 5.3.0 on Ubuntu 14.04. The experiments are run on a 2.6GHz Intel Core i5-3320M processor having 16GB of RAM. Following a method used to tune the GA parameters for the CTOP case is presented.

6.3.1 Parameter tuning for genetic algorithms

As described before, a GA uses a set of parameters to manage its behaviour. A tuning process is essential to find the best parameters for the algorithm. In the literature, one can find methods for parameter tuning. In [114] the general framework for parameter tuning is presented, and various methods are discussed.

In the scope of this work a modified version of the work presented in [115] is used to tune the parameters for the CTOP GA heuristic approach.

The parameter tuning method that is presented in [115] is based on an evolutionary approach itself. It uses a chess rating system for evaluation. Each chromosome in this method is represented by a vector of integer and real values representing the set of parameters. To ensure that the tuned parameters are generalising well, each parameter set is tested against a set of problems. For tuning the presented GA for the CTOP, twelve problem instances are created. They are created by altering the problem size, the vertex distribution, and the number of vehicles. Problem instances of 5x5, 7x7 and 9x9 are used. The vertices are distributed in a grid and a noisy grid form. The noisy grid is created based on the canonical grid and adding some uniform random noise to the position of each vertex. The number of vehicles is 3 and 5. The budget is set to 75% of the maximum budget for all the problems. The rest of the approach is presented in algorithm 6.12.

Algorithm 6.12 CRS-Tuning algorithm

Input: $Problems$
Output: $BestConfiguration$
1: $C \leftarrow$ InitialisePopulation
2: **for** $trial \leftarrow 1, MaxTrials$ **do**
3: **for each** $c \in C$ **do**
4: **for each** $pinProblems$ **do**
5: **for** (**do**$g \in 1, NumGames$)
6: $s \leftarrow GA(c, p)$
7: $S_{c,p,g} \leftarrow s$
8: EvaluateConfigurations(C, S)
9: $parents \leftarrow$ SelectParents(C)
10: $newC \leftarrow$ GenerateNewPop(parents)
11: $C \leftarrow newC$
12: **select** $BestConfiguration$
13: **return** $BestConfiguration$

The first step of the tuning algorithm is to generate a population of chromosomes. Each chromosome is constructed by randomly choosing values for the parameters from a specified range. For the tuning procedure performed for this work, the chromosome population is composed of 100 individuals. Then the algo-

rithm iteratively runs for $MaxTrials$. For this work, $MaxTrials$ are set to ten. Initially, each configuration is used to run each problem $NumGames$ times, and their scores are recorded. Here $NumGames$ is set to ten as well.

Following, the configurations are evaluated using the *Glicko2* chess rating system [116]. The evaluation is performed by each configuration comparing its results against all the other configurations' results and collecting $0, 0.5$ and 1 points for loss, draw and win respectively. These points, as well as the opponents rating, are used to update the fitness of each configuration. In [115] the evaluation is performed based only on the scores of the games. For the tuning performed here a bonus of 0.1 is given to the faster solution, while the slower solution gets a penalty of -0.1. This is done to differentiate the cases where two solutions are equally good utility-wise, but one is slower to compute than the other.

The next step involves selecting the parents that will generate the next generation of configurations. The ten best configurations are selected as parents. In addition to them, any other parents that performed close to the first ten are selected as well. The choice is performed based on their chess rating intervals. The parents have a maximum size of half the total population and are a portion of the new population. The rest of the population is generated by applying uniform crossover and mutation operations with 50% and 80% probability respectively, as suggested by [115]. Finally, the best configuration is selected and returned.

6.4 Reuslts

The first set of results is presented in [110] and compares the performance of the proposed GA heuristic with the MIQP solver for the COP. The GA parameters are presented in table 6.1 and are set in an empirically way. The heuristic is evaluated by comparing the expected accumulated utility and the planning time against the exact solution. The mission performed is the same as the one previously described in section 6.3.

The performance evaluation of the proposed heuristic method for the COP required several tests to be run. The cost of movement is set to be directly proportional to the distance that has to be travelled. Moreover, the vehicle incurs a fixed sensing cost of one unit for each inspection point that was visited. As mentioned in 2.2 the correlation between two points is represented by the kernel

Table 6.1. Genetic algorithm parameters

Parameter	Values
$MaxGeneration$	50
$PopSize$	200
$TourSize$	3
$NumCrossovers$	20
$PopMut$	50
$CrMut$	10
$AddProb$	0.9

function described in (2.8). For this work, the value of l is set to 2. This kernel function is used to calculate a weight for equation (6.1). An example of the maximum correlation range can be seen in figure 6.4.

The tests that are run involve two different runs of the MIQP solver. The first is performed with 1% optimality gap, meaning that the output value is at most 1% different from the optimal. The second is performed using a 5% gap. These are used to limit the execution time of the MIQP to reasonable times. Additionally, there is a maximum allowed amount of time that the solver is allowed to run, and it is set to 36000 seconds.

Regarding the GA, 1000 trials are executed for each size and energy budget. This is required since the GA approach is using randomness in the optimisation process. Two consecutive runs of the GA optimisation are not guaranteed to give the same result regarding utility or runtime; therefore, statistical results need to be extracted.

As mentioned previously, a point of evaluation is the expected accumulated utility of each method. To evaluate that five different grid sizes and four different energy budgets are used. The energy budgets are computed by finding the optimal path to traverse all the vertices using an OVRP instance [117]. To that, all the necessary sensing costs for each vertex were added. The required budgets are then calculated by reducing the maximum budged accordingly. The results of the various tests can be seen in table 6.2.

As expected the 1% optimal MIQP instance usually performs better. The heuristic performance can be mostly compared to the 5% MIQP solution. They perform on average close. By examining the standard deviation of the heuristic, it can be seen that it can find solutions close to the 1% optimal given enough

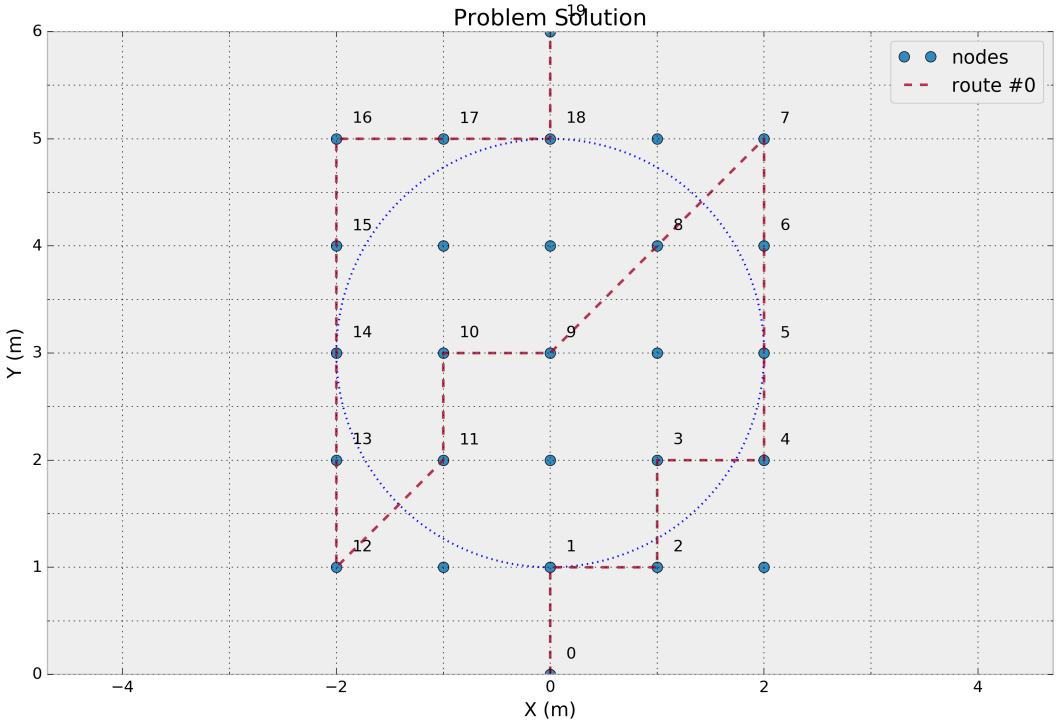
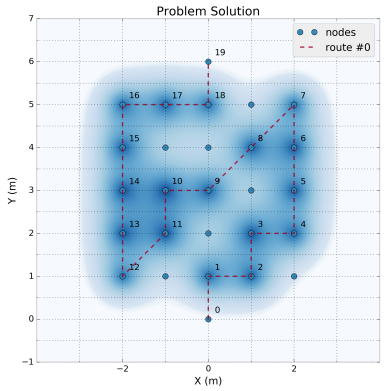


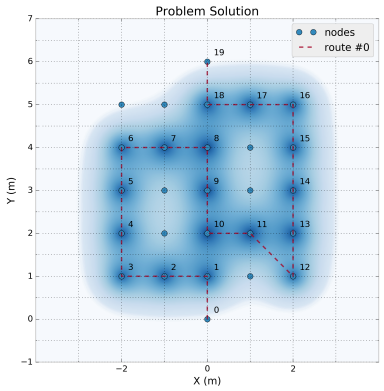
Figure 6.4. Correlation range from inspection point 9. All the non-visited points inside the blue circle contribute to correlation utility to point 9. This represents the information we gain about the other points by sampling that inspection point.

Table 6.2. Utility gained for different grid sizes and budgets

Grid Size	Algorithm	Budget			
		100%	75%	50%	25%
5x5	COP 1%	24.554	**20.771**	**14.725**	**7.108**
	COP 5%	24.377	20.216	14.571	**7.108**
	GA	**25**	20.704	14.486	7.070
	GA St.Dev.	0	0.087	0.287	0.090
6x6	COP 1%	**35.554**	**30.056**	**21.633**	**10.771**
	COP 5%	35.377	30.017	21.364	**10.771**
	GA	**35.554**	29.758	21.210	10.526
	GA St.Dev.	0	0.144	0.281	0.323
7x7	COP 1%	**48.554**	**41.81**	**30.304**	**15.079**
	COP 5%	47.285	41.418	29.849	**15.079**
	GA	**48.554**	40.716	29.013	14.606
	GA St.Dev.	0	0.389	0.422	0.311
8x8	COP 1%	63.377	**54.426**	**39.828**	**20.095**
	COP 5%	62.754	54.165	39.343	**20.095**
	GA	**63.547**	53.194	37.969	19.110
	GA St.Dev.	0.053	0.413	0.549	0.64
9x9	COP 1%	80.377	**69.666**	**51.207**	**25.835**
	COP 5%	79.662	67.804	50.244	**25.835**
	GA	**80.456**	67.624	48.395	24.723
	GA St.Dev.	0.226	0.474	0.791	0.719



(a) MIQP problem solution for a 5x5 grid and energy budget of 38.25 units (75% of maximum). The utility gathered was 20.701.



(b) GA problem solution for a 5x5 grid and energy budget of 38.25 units (75% of maximum). The utility gathered was 20.704.

tries. In figure 6.5a the 1% optimal calculated path for a 5x5 grid with 75% available energy can be seen. The blue colour represents the area coverage due to correlation. The colour is darker closer to the actual sensed points and fades out as the correlation drops. The heuristic solution for the same method can be seen in figure 6.5b. They achieve the same utility. In terms of coverage quality, the MIQP achieves to cover the whole area, while the GA performed better coverage on the central part of the area. It should be noted that the GA gives a solution similar to a standard lawnmower pattern used by vehicles in the field.

The second criterion of evaluation is path planning time. Given that it is required for the vehicle to perform its mission autonomously it is essential that the mission can be planned online and as fast as possible. The results can be seen in table 6.3.

As it can be seen the proposed heuristic performs much better than the 1% MIQP solutions by several orders of magnitude. It also outperforms the 5% gap solution. That is even more visible in larger instances where again it is many times faster. The execution time is ideal for online usage. Since the heuristic is time-efficient, it can be run multiple times in order to find the best solution. This solution, in many cases, will be close to the optimal, but in planning time it will be orders of magnitude faster. The benefits of the heuristic will be even more visible in lower-end or embedded processors used in many autonomous systems.

Before evaluating the heuristic for the CTOP, the genetic algorithm parameters

Table 6.3. Computation time(s) for different grid sizes and budgets

Grid Size	Algorithm	Budget			
		100%	75%	50%	25%
5x5	COP 1%	1.381	5.118	14.981	1.257
	COP 5%	0.514	3.664	7.979	1.076
	GA	**0.221**	**0.166**	**0.127**	**0.101**
	GA St.Dev.	0.03	0.013	0.013	0.014
6x6	COP 1%	34.745	13.118	389.658	22.541
	COP 5%	0.898	1.559	70.983	9.001
	GA	**0.35**	**0.243**	**0.159**	**0.09**
	GA St.Dev.	0.06	0.028	0.017	0.005
7x7	COP 1%	2.42	190.585	2492	560.24
	COP 5%	1.926	2.505	97.181	27.68
	GA	**0.785**	**0.411**	**0.207**	**0.103**
	GA St.Dev.	0.123	0.06	0.022	0.004
8x8	COP 1%	25.15	53.98	28965	5715.92
	COP 5%	11.78	13.91	272.9	97.93
	GA	**1.042**	**0.636**	**0.298**	**0.129**
	GA St.Dev.	0.197	0.077	0.023	0.015
9x9	COP 1%	214.77	518.53	MAX	MAX
	COP 5%	92.247	23.041	865.83	408.81
	GA	**2.038**	**1.1**	**0.452**	**0.165**
	GA St.Dev.	0.361	0.147	0.052	0.017

are tuned with the method described in the previous section. The results of the tuning can be seen in table 6.4. The numbers in the brackets represent the allowable values for each parameter. The two next columns represent the value used with the algorithm for each different population generation method.

The performance of the heuristic method is compared to the optimal solution in two different ways. The first way examines how well the heuristic method performs for different numbers of robots. For that, the maximum budget is allocated to each robot. The maximum budget is calculated by taking the maximum budget required for a single robot to complete the mission and dividing it by the number of robots. The results can be seen in figure 6.6 and Table 6.5.

Figure 6.6 presents statistical results regarding the utility for each size of the robotic team using the GA based planning. It is compared against a solution produced by solving the MIQP problem using the Gurobi solver. The solver was

Table 6.4. Parameter ranges and tuned values for the two different methods

Parameter	Value Range	Random	NNRASP
Population Size	$[25, 50, \ldots, 500]$	300	250
Generations Number	$[5, 10, \ldots, 50]$	40	50
Tournament Size	$[3, 4, \ldots, 10]$	6	5
CX Probability	$[0.0, 0.1, \ldots, 0.9]$	0.7	0.9
Mutation Probability	$[0.0, 0.1, \ldots, 0.9]$	0.6	0.7
Elitist Percentage	$[0.01, 0.02, \ldots, 0.20]$	0.19	0.03

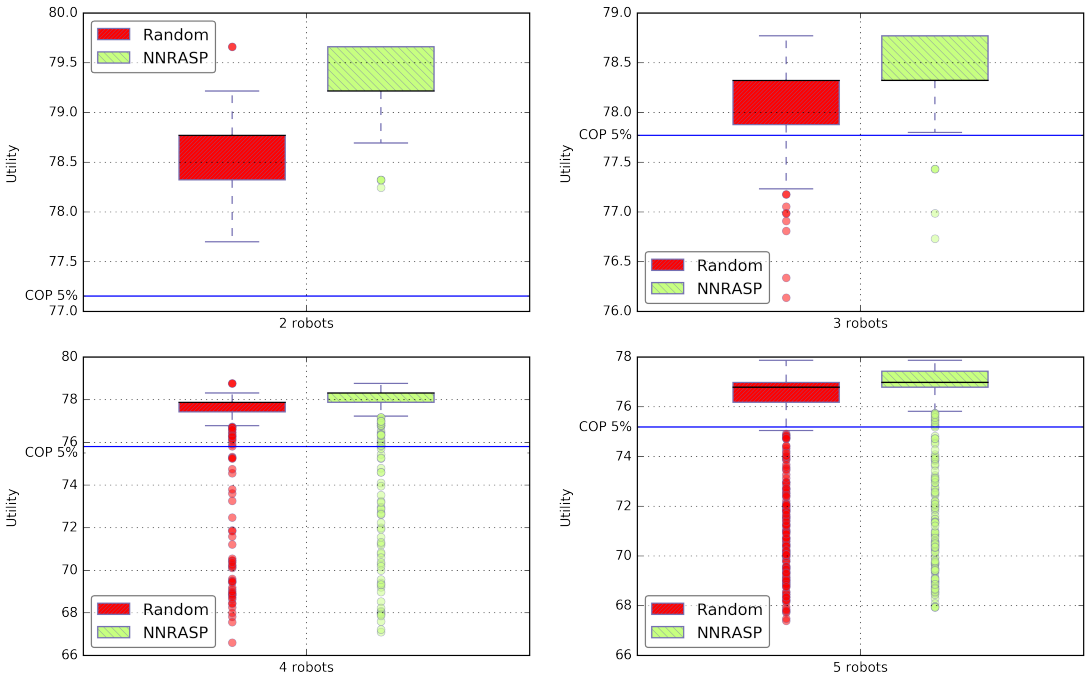


Figure 6.6. Utility box plots for different numbers of robots. For lower number of robots a sophisticated population initialisation method tends to give better results. As the number goes higher the results come closer with a slight superiority of the NNRASP method. The blue line represents the 5% gap optimal solution obtained by solving the MIQP problem.

Table 6.5. Average time(s) for different algorithms and vehicle numbers

Algorithm	Number of Robots			
	2	3	4	5
CTOP 5%	396.36	695.60	2830.41	1474.18
GA-Random	**0.787703**	**0.729135**	**0.666236**	**0.623074**
St.Dev.	0.087695	0.0738167	0.0602966	0.0579891
GA-NNRASP	0.792475	0.844305	0.817855	0.826688
St.Dev.	0.106818	0.120596	0.112062	0.106136

stopped once the found solution was guaranteed to be at most 5% worse than optimal. The horizontal blue line represents this solution. It can be seen that the GA heuristic with the NNRASP generation method outperforms the MIQP solution in all the cases. The random generation method falls a bit behind, with part of the results produced being worse than the 5% optimal.

Comparing the average planning time the GA heuristic heavily outperforms the MIQP exact method, as it can be seen in Table 6.5. It can be seen that the time required for the optimal solution is in the best case 500 times higher. The GA-Random method performed slightly better than the GA-NNRASP. Example paths for a three-robot team can be seen in figures 6.8a, 6.8b and 6.8c. The first figure shows the 1% optimal path calculated by the MIQP solver, while the second is a 5% optimal solution used for comparison with the GA. The heuristic solution can be seen in the third image. It is obvious that the solution quality of the GA is far better than the 5% optimal.

The second performance metric examines how the heuristic method performs in cases where the budget is limited. For this, the number of vehicles is chosen to be three and is kept static. Then the budget of each vehicle is reduced by a specific percentage. Figure 6.7 and table 6.6 present the results for this metric.

In figure 6.7 the utility gained for each budget is shown. It can be seen that the GA method performs at least equally good as the 5% exact method. That happened only in the case where the robots had 25% of the maximum energy. In all the other cases the GA performed better. Comparing the two different population generation methods, one can see that the more sophisticated approach gives statistically better results. The random method performs close to the 5% optimal.

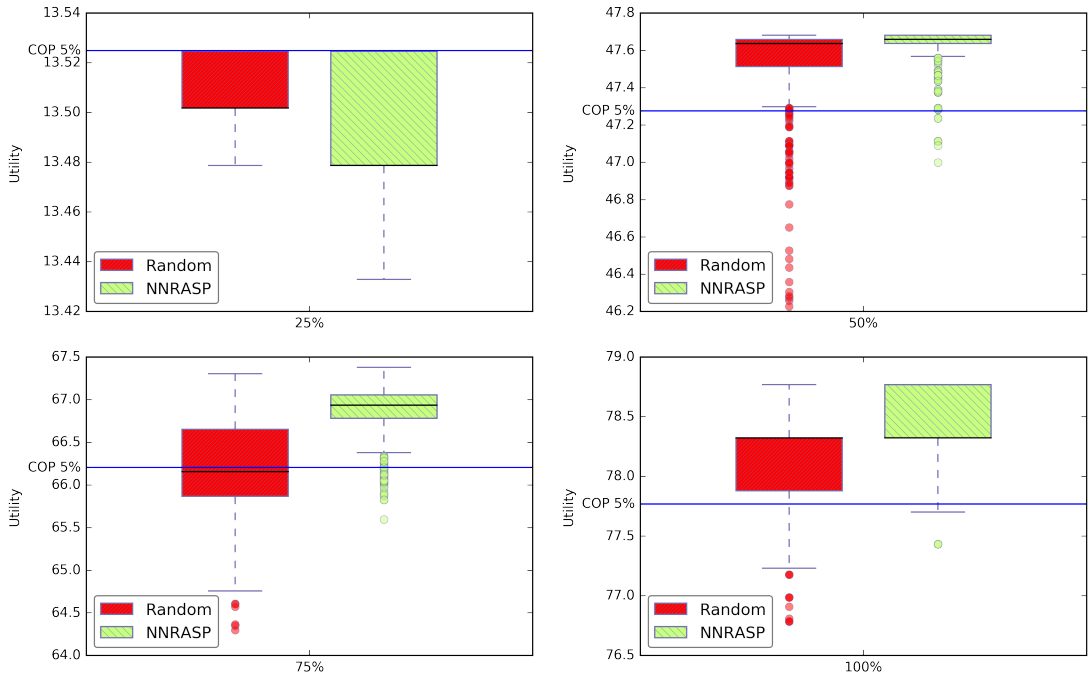
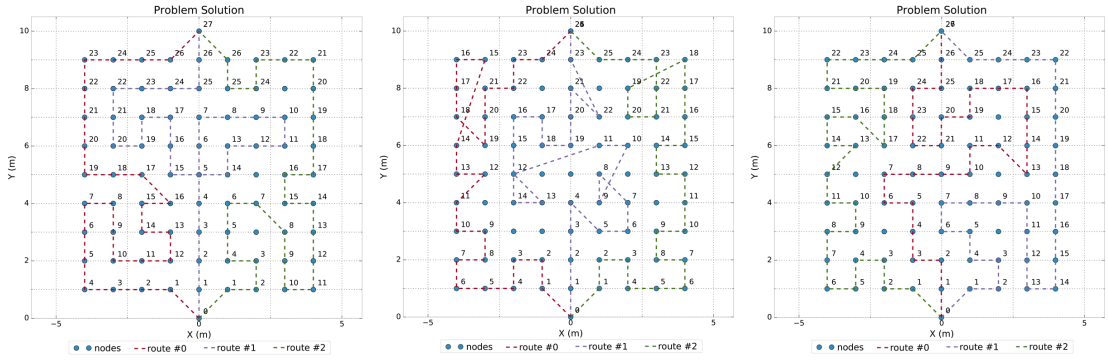


Figure 6.7. Utility box plots for three robots having different budget. For lower budgets a sophisticated population initialisation performs equally good with the random one. As the budget goes higher the results show the benefits of the NNRASP method. The blue line represents the 5% gap optimal solution obtained by solving the MIQP problem.



(a) 1% optimal MIQP solution. **(b)** 5% optimal MIQP solution. **(c)** Genetic algorithm based heuristic solution.

Figure 6.8. Comparison of the paths generated for three vehicles having full budget. In 6.8a the gathered utility is 79.662 and the time to calculate is 5847.81 seconds. In 6.8b the gathered utility is 76.433 and the time to calculate is 690.74 seconds. In 6.8c the gathered utility is 78.770 and the time to calculate is 0.8597 seconds.

Table 6.6. Average time(s) for different algorithms and different budgets

Algorithm	Budget			
	100%	75%	50%	25%
CTOP 5%	695.60	267.94	3611.76	3764.12
GA-RANDOM	**0.721606**	**0.625391**	**0.362355**	**0.141678**
St.Dev.	0.0742465	0.0607859	0.0346406	0.0147053
GA-NNRASP	0.837791	0.763285	0.389283	0.165087
St.Dev.	0.120398	0.102063	0.0483728	0.0244343

In table 6.6 the timing results of the limited budget experiments are presented. As with all the previous cases, the GA-Random method performed better. In the worst case, it was more than 300 times faster than the 5% MIQP solution. The GA with NNRASP population generation method constantly performed a bit slower than the Random generation method.

6.5 Summary

This chapter presented the concept of a sensing mission under budget constraints along with solution methods depending on the mission characteristics and the number of vehicles. Exact solutions for the COP and CTOP problems were presented using MIQP formulations. GA based heuristics were presented to achieve close to optimal online execution. Simulated sampling missions of various sizes were used to verify the correctness of the approaches. For both the single and the multi-vehicle approaches the heuristics produced solutions that were close to optimal requiring only a fraction of the computational resources compared with the MIQP solution. This makes the heuristic methods applicable in online settings. The next chapter will present a novel approach for long range mission optimisation that combines methods from both chapters 5 and 6.

Chapter 7

Large scale sensing missions

In chapter 6 the problem of optimising sensing missions was discussed. Solution methods for single and multiple vehicle teams operating under budget constraints were presented. These methods allowed the online optimisation of the sensing missions.

The current chapter studies larger sensing missions than the ones presented in chapter 6. In particular, it studies missions where multiple regions need to be visited for samples to be collected. Given the problem size only multiple vehicle teams are considered. As in the single region problems, the multi-region problem is also studied under budget constraints.

In the rest of the chapter, the problem will be described in detail, and a proposed solution method will be provided. The proposed method is tested against the multi-vehicle method and results will be accordingly presented. Finally, conclusions on the proposed approach will be discussed.

7.1 Problem setting

The problem addressed in this chapter is the one of multi-region sampling, as briefly mentioned before. In this problem multiple regions of interest, for example, pollution spots, need to be sampled over a larger area. Each of those regions can vary in size, and their size may change over time. An example of such a problem can be seen in figure 7.1.

For solving this problem one or multiple robots can be employed. Given the size of the problem, a multi-vehicle approach would provide robustness and speed to the problem solution, as regions can be sampled in parallel and vehicle failures

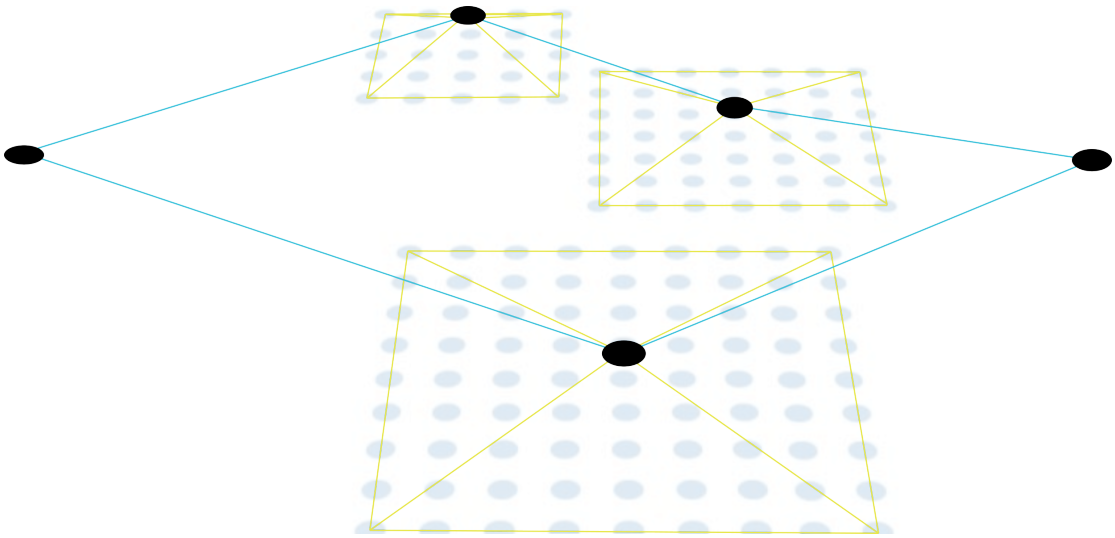


Figure 7.1. 2-level mission performed by two vehicles. The high-level optimisation optimises the way the regions are visited. This is represented by the black vertices connected by blue lines. The grey vertices are the sampling points that need to be visited. There a solution to the COP is found using the GA heuristic.

would not compromise the execution. As with the problem described in chapter 6, the robots will be operating in a dynamic environment. This can have an effect on the available resources for completing the mission. Therefore, this problem is also studied under the prism of resource or budget constraints. As with the problems studied in chapter 5 the regions to be sampled can be known at the beginning of the mission or be discovered dynamically during mission execution. In any case, a heterogeneous team would be ideal for solving the studied problem. Specifically, there could be one or more vehicles discovering regions that need to be sampled while other team members would proceed to collect the samples from those regions.

The current state of the art, presented in [118], focuses on a single region that needs to be sampled. Though it can be directly applied to the multi-region problem, its solution procedure may prevent it from being applied in real-world applications. The solution method of [118] is a centralised method that produces the path that each vehicle of the team should follow in order to maximise the sampling output. This centralised approach makes it reliant on communicating all the relevant information to a single processing node, which in turn will have to send the results to the vehicles in order to execute the mission. This approach can

be hard to apply in cases where communication is unreliable and of low bandwidth, such as an underwater environment. In addition, the current method can not be applied in a decentralised manner, as its randomised search nature would create a different global plan each time it would be run, creating inconsistencies that could potentially reduce the outcome of the mission. To overcome the aforementioned issues, a new method that combines the optimal allocation described in chapter 5 with the close to optimal trajectory generation of chapter 6 is presented.

7.2 Two level optimisation

As already mentioned, a new method is proposed to allow optimised sensing missions that require samples to be collected over multiple regions. The proposed approach consists of two levels of optimisation.

The first level of optimisation aims at producing a consistent global plan among all the agents. This global plan represents which regions should be sampled by each vehicle, as well as the order to visit those regions. The solution for this plan is obtained by solving an MDmTSP instance as the one presented in chapter 5. Each vehicle is represented as a depot of the problem, having one extra depot as the final point where all the vehicles should reach after the completion of the mission. The rest of the nodes represent the centres of the regions that need to be sampled. The MDmTSP solution will give a schedule for each vehicle. Each schedule will have one or more regions to be visited, in addition to the initial position of the vehicle and the final position. In case that there are fewer regions than vehicles, the problem is infeasible. In such cases, an exhaustive search method can produce an optimal solution.

This type of optimisation can be performed in a distributed fashion, as the optimisation method is deterministic and is guaranteed to give the same result each time that is run with the same parameters. This behaviour was also verified in section 5.5.2 where multiple vehicles executed an inspection mission in a distributed manner even in the presence of communication errors.

7.3 Budget calculation

One of the essential aspects when optimising large scale sensing missions under budget constraints is how to distribute the budget among the regions that need to be sampled. This section will describe the process of calculating the available budget and how it is then allocated in each region.

As mentioned in section 7.2 the optimisation process is performed in two levels. In the first level, the optimal plan on how to visit the sampling regions is generated. This optimal plan is computed by representing each sampling region by its central point. The result of the optimisation is a number of paths equal to the number of vehicles. Each of these paths is represented by a vector containing a number of these central points in the order that they should be visited.

The next step is to compute the travelling cost spent while moving between the regions. The path cost that is computed by the high-level optimisation does not represent the actual travelling cost between the regions. This is because it is computed based on the centre points of the regions. In reality, the robot will collect samples from the whole region so it will move from the side of one region to the other. This means that the actual travelling cost will be less and more budget will be available for sampling. Therefore, each robot must know the exact cost to be able to calculate the amount of budget that can be allocated to sampling.

In figure 7.2 one can see two sampling regions. Their connected centres are what the high-level optimisation is producing. As can be seen, the line connecting the centres is intersecting with the sides of the two regions. One can calculate the intersection point and find the closest sampling point to that. This way the starting and ending points of the sampling regions can be calculated. It must be mentioned that this is applied to rectangular shaped regions that are axis aligned. Such regions are in the scope of this work.

As briefly mentioned in the previous paragraph, the starting point will be one of the sampling vertices of the region. Specifically, this vertex will be on the side that is intersected by the line that connects the centre of the region with the centre of the previous region as given by the top level planning. To calculate the starting point of a region one can employ simple algebraic operations. Firstly, the gradient of the line m_l connecting the two centres must be calculated. This can be done by using the gradient equation as shown in equation (7.1). The second step involves

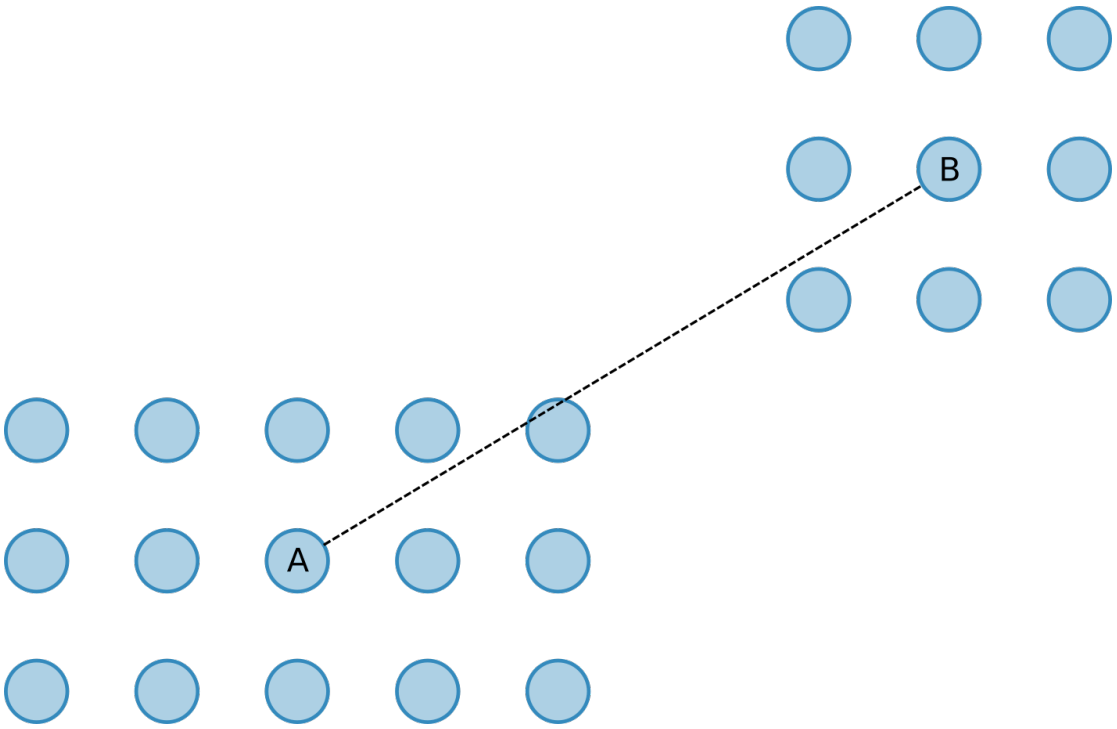


Figure 7.2. Two sampling regions and a line that connects their centres. This line is the output of the top level planning. While the mission is executed the vehicle will exit region A from a sampling point and will enter region B at a sampling point.

computing the gradient of the diagonals m_d^1 and m_d^2 of the region. This is done using the same equation and the region's corner points. The region along with the diagonals and the line connecting the centre can be seen in figure 7.3a.

$$m = \frac{\Delta y}{\Delta x} = \frac{y_B - y_A}{x_B - x_A} \tag{7.1}$$

To find which side of the region is intersected by the line one can compare the absolute value of the gradient m_l with the absolute values of m_d^1 and m_d^2. In the case of rectangular shaped regions studied here one can assume that $\mid m_d^1 \mid = \mid m_d^2 \mid$. This comparison gives three intersection cases. In the first case, the absolute value of the centre line gradient is less than the absolute value of the diagonal gradient. In that case, it intersects either the front or the back of the region. To determine which side is intersected one can use the Δx component of the gradient. If $\Delta x > 0$ it intersects the back of the region, while when $\Delta x < 0$ it intersects the front. The second case is defined when $\mid m_l \mid > \mid m_d \mid$. This denotes that the line connecting

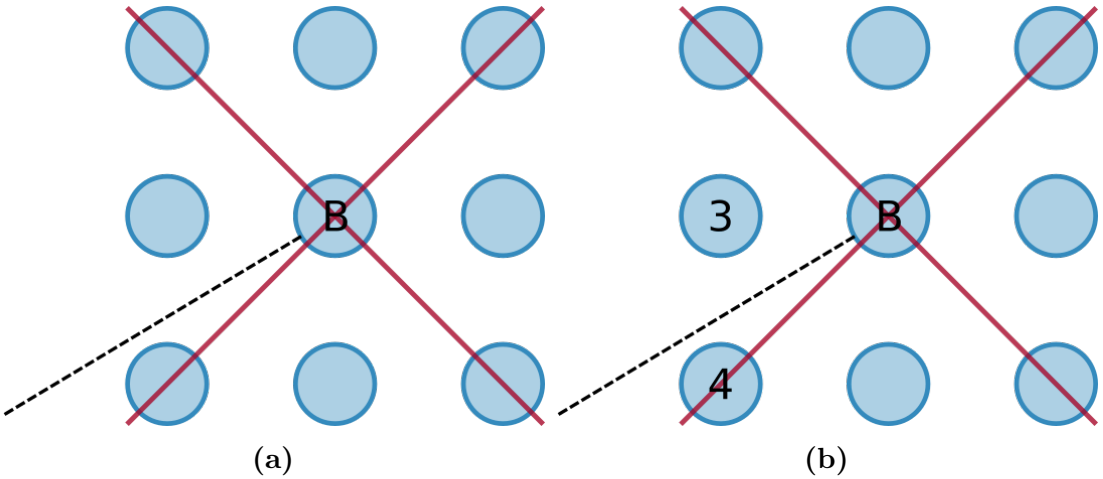


Figure 7.3. Calculation of the starting sampling point. In figure 7.3a one can see the red diagonals of the area that are used to find the intersection with the line connected to the centre. The candidate starting points for region B can be seen in figure 7.3b. The closest points to the line are marked by numbers 3 and 4.

the centres intersects the top or the bottom of the region. The side is determined by the sign of the Δy component of the gradient. If it is positive, it intersects the bottom side, and if it is negative, the top is intersected. The final case is when m_l is equal to m_d. Then the intersection is one of the corners of the region. Which corner can be defined by the values of Δx and Δy of the centre connecting line gradient.

In the example case of figure 7.2, the line connecting the centres has a gradient with an absolute value smaller than the absolute value of the gradient of the diagonals as seen in figure 7.3a. This means that it would intersect the back or the front of the region. Given that Δx is positive, one can find that the back side is intersected.

The next step is to find the intersection point between the line and the side. This can be done by solving the linear equation system of the two lines. To determine which sampling points are the closest to the intersection point one can use the floor and ceiling functions. In the example of figure 7.3b the intersection is on the back of the region. In both the sampling points the value of x is the same. By applying the floor and ceiling functions to the y coordinate of the intersection point, one can get the y coordinates of the closest sampling points, namely points three and four.

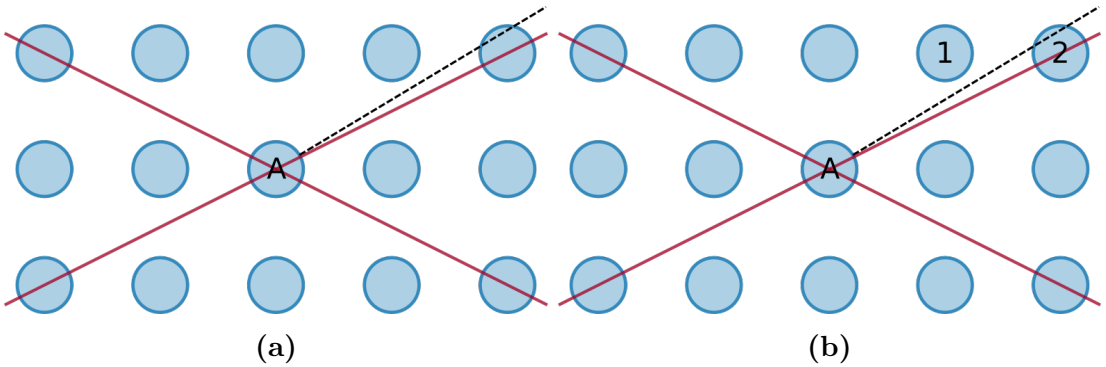


Figure 7.4. Calculation of the finishing point. The diagonals that help identify the side of the region the point is are can be seen in figure 7.4a as red lines. The candidates of the finishing point are vertices numbered 1 and 2 and can be seen in figure 7.4b.

An equivalent method can be followed for finding the end point of the region. The only difference is how the decision is made on the intersecting side. When the front or the back of the region is intersected the Δx is still used. However in this case, when $\Delta x > 0$ then the front is intersected. Accordingly when $\Delta x < 0$ the back is intersected. Likewise, for the top-bottom case, when $\Delta y > 0$ the top side is intersected. For $\Delta y < 0$ the line intersects the bottom side. In the example presented in figure 7.4a one can see that the absolute value of the gradient of the line is higher than the absolute value of the gradient of the diagonals. Therefore, the intersection happens on either the top or the bottom of the region. Since $\Delta y > 0$ the top side is intersected. Calculating the closest sampling points to the intersection gives points one and two as seen in figure 7.4b.

To find the path that connects the two regions, and the starting and ending points respectively, an exhaustive search is possible. In figure 7.5 the four candidates can be seen on the edges of the regions. An exhaustive search would involve four comparisons. It can be seen that the closest points are two and four. Thus, point two will be the ending point for region A, while point four will be the starting point for region B. The cost of travelling from region A to region B will be the length of the path from point two to point four. It can be seen that the cost is much lower than the one used from the top level planning.

When all the starting and ending points for each region are found, the actual cost of travelling can be calculated. This cost is then subtracted from the available budget. The remaining budget is what can be used for sampling. The remaining

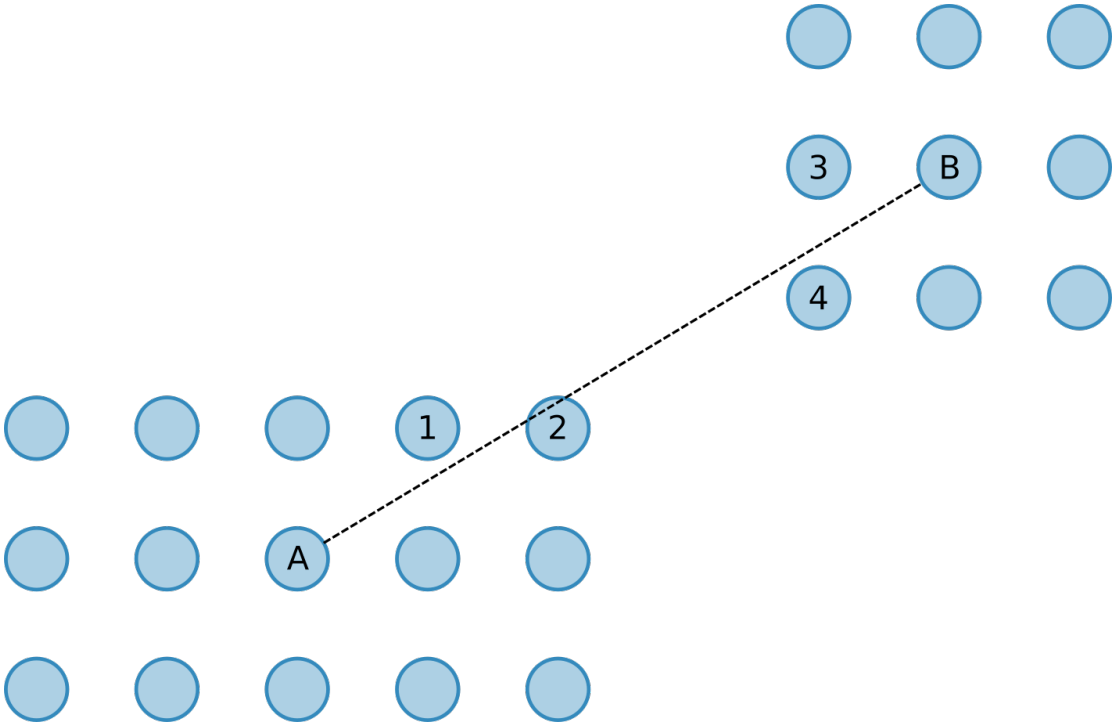


Figure 7.5. The ending (1,2) and starting (3,4) candidates for the two sampling regions. With a simple search strategy, one can find the closest distance using at most four comparisons. In this example it is obvious that the shortest distance is between vertices 2 and 4, making them the finishing and starting vertices of regions A and B respectively.

budget is allocated proportionally to the sampling regions. Each region gets a percentage of the budget based on its size. The total amount of sampling points is calculated. A percentage is then computed for each region by dividing the number of sampling points of the region by the total amount of points. Multiplying this with the remaining budget gives the budget that can be spent on sampling in each region.

7.4 Experimental setup

To test the effectiveness of the two-level planning method an experimental procedure needs to be employed. The testing procedure consists of a long-range sampling mission. In that mission, multiple regions need to be sampled by a team of vehicles. As in the results presented in chapter 6, the evaluation is performed on two criteria, namely the utility of the plan of each method and the time it took to generate the plan.

To evaluate the methods in a generalised manner, random missions are used. In these missions, the sampling regions are randomly generated over a large area. Evaluation is performed over a large number of missions, and statistical results are extracted. This section presents the methods used to generate the missions and how the results are collected.

7.4.1 Random mission generation

As already mentioned, the two methods are tested using randomly generated missions. In this mission sampling regions are randomly generated over a larger area. These sampling regions can be described by their centre point, their width and their length.

In general, a random point pattern in a d-dimensional space is described by a spatial point process[119]. Usually in applications $d = 1$, $d = 2$ or $d = 3$. One-dimensional processes can be used to model event occurrences over time, while two or three-dimensional processes can be used to model event occurrences over space. Spatial processes have a wide variety of application over different disciplines of science. For example in statistical ecology [120], [121] they are used to model locations of trees or bird nests. In astrostatistics [122] they model the positions of stars and galaxies, while in material science [123] they model defects in silicon crystal wafers. Finally, in statistical epidemiology, they model the addresses of people diagnosed with a specific disease. [124].

A particular case of a spatial point process is the spatial Poisson process. In the Poisson process, the number of points in an area A is following a Poisson distribution with mean $\beta\lambda_2(A)$. Here β is the intensity of the process governing the number of points per unit area, while $\lambda_2(A)$ is the area size of A. In the case studied by this work, the Poisson process is used to generate the centre points of the sampling regions that are spread over the larger area that is studied. The process that is used to generate a mission can be seen in algorithm 7.1.

The algorithm starts by calculating the percentage of the area that is covered by regions that need to be sampled. This can be expressed as the total number of sampling points. This is done by calculating the size of the area and multiplying it with the coverage percentage of it. The approach followed here assumes that the large area can be represented as a grid where sampling points can be placed.

Algorithm 7.1 Random Problem Generation

Input: $AreaWidth, AreaLength, AverageRegionSize, CoveragePercentage, MinRegionsNumber$

Output: $SamplingRegions$

1: $NumNodes \leftarrow AreaLength * AreaWidth * CoveragePercentage$
2: $MeanRegionNum \leftarrow NumNodes/AverageRegionSize$
3: $NumRegions \leftarrow$ SamplePoisson($MeanRegionNum$)
4: **if** $NumRegions < MinRegionsNumber$ **then**
5: $NumRegions \leftarrow MinRegionsNumber$
6: **while** $NumRegions > 0$ **do**
7: $RegionCentre_x \leftarrow$ SampleUniform($1, AreaWidth$)
8: $RegionCentre_y \leftarrow$ SampleUniform($1, AreaLength$)
9: $RegionSize \leftarrow$ SampleNormal($AverageRegionSize, 1$)
10: $RegionSize \leftarrow$ Round($RegionSize$)
11: $NSA \leftarrow$ Samplingregion($RegionCentre_x, RegionCentre_y, RegionSize$)
12: **if** $\neg$Overlaps($NSA, Samplingregions$) **then**
13: Insert($Samplingregions, NSA$)
14: $NumRegions \leftarrow NumRegions - 1$
15: **return** $Samplingregions$

In the next step, the mean number of sampling regions is computed by dividing the number of nodes by the average sampling region size. Following, the actual number of sampling regions present in the problem is sampled from a Poisson distribution with $\mu = MeanRegionNum$. If the number of sampling regions is less than a user-defined threshold, then it is replaced by that threshold. This was done to avoid cases where the number of regions is less than the agents making the whole problem trivial to solve.

After the number of sampling regions is decided a repetitive process to generate them begins. The first step in that process requires the centre of the sampling region to be randomly generated. This is performed by sampling from a uniform distribution over the width and the length of the larger area. Next, the sampling region size is determined by randomly generating a size from a normal distribution with $\mu = AverageRegionSize$ and $\sigma = 1$. Finally, if the region is not overlapping with any of the already generated regions is inserted to the regions set. After all the regions are generated they are returned so that the mission can be optimised.

7.4.2 Experimental procedure

Given that both methods use stochastic optimisation processes, only statistical results regarding their performance can be collected. Therefore, multiple missions are generated using the procedure described in the previous section. The evaluation is performed in terms of utility and time to compute a solution using three different tests. The first test evaluates the scaling of the method to various percentages of area coverage. The number of robots is set to three and the maximum budget is allocated. Then the coverage percentage of the area is having values of 10%, 15% and 20%. The second test evaluates the performance in cases of a limited budget. The number of robots is still three, while the coverage percentage is set to 10%. Then the maximum available budget takes values of 100%, 75% and 50% of the maximum budget. The final test examines the scaling of the method to various sizes of robot teams. In that case, the maximum budget is allocated, and the area coverage is still at 10%. The number of robots is then getting values of 2, 3 or 4 robots.

As already seen the generation method creates random missions that differ in size. Consequently, a direct comparison of the utility and time cannot be performed. For that, the metrics are normalised on the number of nodes. So a comparison is performed on the utility per node and the computation time per node. Another issue that arises because of the random mission generation is the one of the maximum budget available for performing the mission. To calculate the maximum budget, a simple heuristic is used. In that heuristic, the cost of sampling each region is calculated using an OVRP instance as described in the results of the COP in chapter 6. Then the high-level optimisation is performed, and the cost of travelling to the centres of all the regions is calculated. Finally, the maximum budget for the team is computing by adding all these costs together. This budget is then divided equally among the agents.

7.5 Results

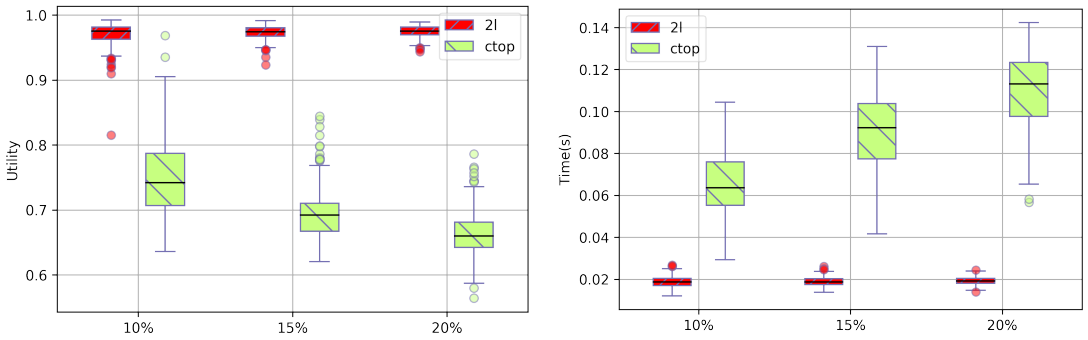
This section presents the results of the experimental procedure presented before. For each testing case, 500 experiments were run, and statistical results are extracted. The parameters used to generate the random problems are shown in

table 7.1. The parameters for the genetic algorithm heuristics of the COP and CTOP are the same as the ones used for obtaining the results of 6.

Table 7.1. Genetic algorithm parameters

Parameter	Values
$AreaWidth$	100
$AreaLength$	100
$AverageRegionSize$	9
$MinRegionsNumber$	$2 * NumRobots$

The first experiment is checking how is the 2-level method performing compared to the CTOP for different problem sizes. For this, the maximum budget was allocated, and three robots were used. Results for this can be seen in figures 7.6a and 7.6b.



(a) Utility boxplots for different area coverage percentages. The 2-level method performs constantly close to the optimal. The CTOP performance is worse and reduces as the coverage increases.

(b) Time boxplots for different area coverage percentages. The 2-level has a constant behaviour that is always better than the CTOP. The CTOP requires more time as the coverage increases.

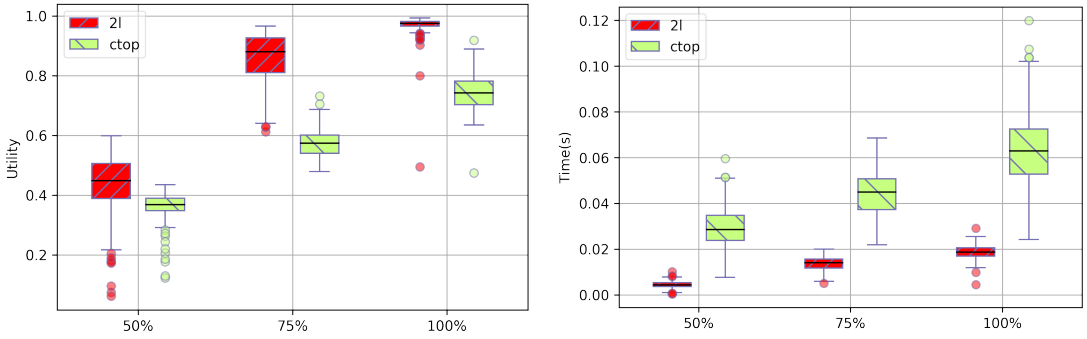
Figure 7.6. Utility and time boxplots for different area coverage percentages. Three vehicles were used and the maximum budget was available.

Figure 7.6a presents the utility obtained per sampling point. It can be seen that the 2-level method consistently performs close to the optimal not getting influenced by the size. On the other hand, the CTOP method performs constantly worse than the 2-level method. In addition, the CTOP performance degrades as the area size increases. A possible cause of that is that it is stuck into local optima and does not have enough time to find a solution that will escape it.

Regarding the time required to find a solution, the 2-level behaviour performs

really well. It can be seen in figure 7.6b that it requires constant time per sampling point, while the 0.02 seconds per point make it applicable online. It additionally outperforms the CTOP by at least a factor of 3. The CTOP is consistently performing worse, and its computational complexity appears to be linear to the coverage percentage.

The second experiment examines how the 2-level method performs in cases of a reduced budget. The number of vehicles is still three, while 10% of the total area is sampled. Results regarding the utility gathered and the total time to compute a plan are shown in figures 7.7a and 7.7b respectively.



(a) Utility boxplots for different budgets. The 2-level method performs better than the CTOP in most cases. The CTOP performance is usually worse.

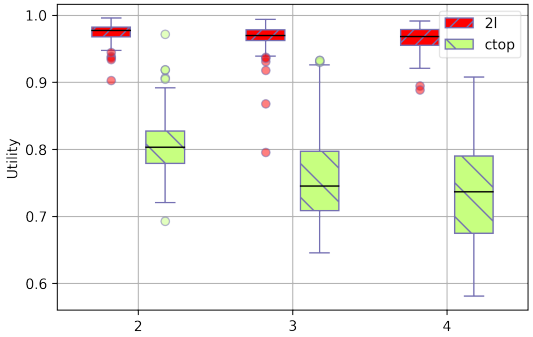
(b) Time boxplots for different budgets. For both cases the time requirements increase as the budget increases. The 2-level method performs better than the CTOP.

Figure 7.7. Utility and time boxplots for different budgets. Three vehicles were used and the area coverage was 10%.

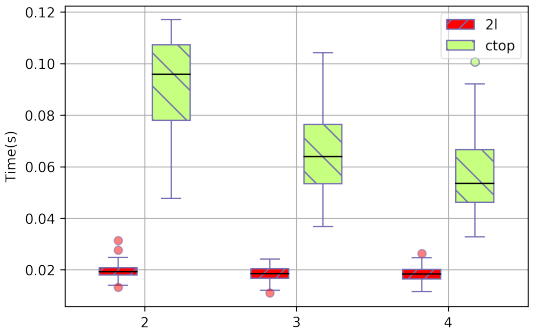
Regarding the gathered utility per sampling point, the 2-level approach still outperforms the CTOP as seen in figure 7.7a. In cases of 100% and 75% of the budget, it clearly performs better. In the case where 50% of the budget is available, the performance of the two methods converges with the 2-level performing better on 75% of the cases. In some outlier cases, it performs much worse. This can be caused by an uneven distribution of points among the vehicles. Since the high-level optimisation only considers the centres of the sampling regions and not their size, there can be cases where one vehicle gets the majority of the points and will not have enough budget to visit most of them. To solve such a case, the load of each vehicle can be checked, and a heuristic can be employed to improve the situation. For example, it can try to split one area into smaller areas or try to assign the area to another vehicle.

In figure 7.7b the collected results regarding the time are presented. The 2-level method still outperforms the CTOP by again at least a factor of 3. It should be noted that both the methods appear to scale linearly with the amount of budget.

The final experiment considers the performance of the 2-level method in teams of various sizes. For that, the budget is set to the maximum available, and the coverage is kept to 10%. The vehicle number varies from 2 to 4. The results can be seen in figures 7.8a and 7.8b.



(a) Utility boxplots for different team sizes. The 2-level method performs constantly close to optimal and better than the CTOP. The CTOP performance worse and degrades as the team size increases.



(b) Time boxplots for different team sizes. The 2-level has a constant behaviour that is always better than the CTOP. The CTOP requires less time as the team size increases.

Figure 7.8. Utility and time boxplots for different team sizes. The area coverage was 10% and the maximum budget was available.

The results for the gathered utility can be seen in figure 7.8a. The 2-level approach clearly outperforms the CTOP in all the cases. Again it has a performance close to the optimal as each of the sampled vertices contributes almost 1 unit of reward. The CTOP performance appears to be linearly decreasing to the number of vehicles. This is probably due to the local minima that the solution is stuck to as it tries to optimise more paths with less budget per path.

The time requirement results can be seen in figure 7.8b. The 2-level proposed approach still has constant time requirements with the number of vehicles. In addition, it performs at least 3 times better than the CTOP. The time requirements per visited sampling point are lower as the number of vehicles increases. One can hypothesise that as the number of vehicles increases, the budget per vehicle is reduced, and the smaller paths reduce the computational efforts of the GA operations.

7.6 Summary

This chapter presented an approach for optimising long range sampling missions where many regions need to be sampled over a larger area. It combines the methods presented in chapters 5 and 6 allowing multi-vehicle sampling missions to be performed. Compared with the CTOP heuristic that is already used to optimise sampling missions for multiple teams it performs better both in terms of utility and computational time. It can achieve close to optimal performance while having constant execution time in many cases. The next chapter will conclude this work and provide recommendations for potential directions for future research on the topic.

Chapter 8

Conclusion and Future Work

This work presented novel approaches for optimising robotic missions executed by a team of robots that operates under communication or budget constraints. The approaches can extend the capabilities of a robotic team as they can allow missions to be executed faster or within specific resource limits. The presented methods were implemented and tested both in simulation and on real vehicles. The acquired results validate the approaches and highlight the benefits of their usage in various scenarios. Inspection task optimisation can reduce the execution time of a mission allowing more missions to be performed. Sampling task optimisation can maximise the information gathering especially in cases where the budget is constrained.

The proposed approaches address the three main questions presented at the beginning of this work. Namely, they are addressing the following:

- How to address the multi-robot task allocation problem under communication constraints?
- How to optimise mission performance for sampling tasks under budget constraints in an online manner?

The first question is addressed by using an information-sharing framework that is robust to communication failures. The use of such a framework ensures the information flow among the members of the team. By maintaining a consistent view of the world, the team members can plan their actions accordingly allowing for optimised mission execution. To optimise the team's behaviour during inspection tasks, methods from the field of optimisation were accordingly modified and

integrated with the aforementioned information sharing framework. Their combination allowed for a decentralised approach to the MRTA that can have optimal performance even in high communication error rates.

Regarding the second question, there were approaches in the literature that addressed the sampling problem under the presence of budget constraints. Their computational complexity made their online use prohibitive, rendering them inapplicable in dynamic environments. The introduction of heuristic solutions allowed the methods to be run online. These heuristic solutions performed close to optimal using a fraction of the computational resources. This can allow the mission to be adapted online taking into account all the new information regarding the environment and the vehicles budget consumption.

Given the results, it has been shown that it is possible to optimise a robotic mission despite and constraints. The information sharing framework can address failures in communication in a reliable way. The MRTA problem can be solved reliably and close to optimal by using such a communication framework. The sampling mission problem can be addressed online even in cases where processing power is low. Combining the MRTA for the inspection task optimisation with the online heuristics for the sampling tasks it was possible to propose a novel method for optimising long-range sensing missions. These advancements allow the robotic team to autonomously plan and perform missions that can be adapted on the fly in case of disturbances allowing prolonged deployments with close to optimal results.

8.1 Major findings

The experimental validation of the proposed approaches gave enough data to show the benefits of them. Following is a list highlighting each one of them:

1. The presence of reliable communications is important in solving the MRTA problem. This is especially true if a decentralised approach is attempted. Having outdated information regarding the status of the team can lead to suboptimal performance. Therefore, combining a framework that ensures communication reliability with optimisation techniques increase the probability of optimal performance from the team. The study of [6] also confirmed this.

2. Heuristics can be valuable assistants to solve hard problems. In the real world, it is usually preferable to find a good enough solution fast and update it during execution as more data is available. This can lead to better results than cases where the optimal solution cannot be computed online due to computation costs. Preliminary results on the topic can be found in [125].

3. The combination of various methods is required for prolonged autonomy to be possible. Optimisation can be achieved in various levels leading to close to optimal decentralised applications. Close to optimal performance can, in turn, increase the output of missions making the robotic teams an even more valuable asset.

Despite focusing the application to underwater scenarios, it is strongly believed that this research is relevant to any branch of field or service robotics. Be it surface, land or airborne vehicles optimised performance is a crucial aspect in their success. Robust and optimised behaviour can, in turn, lead to a broader adoption form other fields of scientific research giving even more benefits to the scientific community and society in general.

8.2 Future work

Despite the findings and proved performance of the proposed approaches, various aspects have been identified as targets for potential improvement and extensions. They are related to both the information sharing framework, the MRTA approaches and the sampling mission optimisation. Following is a list highlighting some of them:

1. The current implementation of the information sharing framework can be hard to scale for a large team. It is based on a TDMA MAC approach limiting the effective bandwidth quite substantially. A newer networking architecture would benefit the incorporation of such frameworks in larger teams. Initial exploration on the subject was performed in [126] and showed the benefits of other MAC protocols. Further investigation is required so that a new framework can be developed.

2. Currently, the information sharing framework has static definitions regarding which information is to be shared and its priority. In real life, information

value changes based on the evolution of the mission. Given that the information sharing network has access to all the semantic information present on a vehicle it can use it to infer which information is relevant to which vehicles and if it is worthy to transmit that piece of information. This would reduce the bandwidth requirements as only important information would be transmitted.

3. Regarding the MRTA approach that is proposed in this work it would be beneficial to compare it against other optimisation methods. For example, a *MinMax* approach that would minimise the maximum effort of an agent may yield better results. Additionally, the approach should be extended towards heterogeneous and time-extended tasks. The current scheme considers only tasks that are the same and have the same value and requirements to be performed. In the real world, on the other hand, tasks usually have quite individual characteristics making them hard to be solvable equally good by each available robot. This would extend the applicability of the approach in more domains.

4. Regarding the sampling mission optimisation the current approach assumes that the characteristics of the area to be sampled are known beforehand. In many real-world situations, this is not the case. An initial exploration strategy could be devised in order to learn the characteristics of the phenomenon and then a full optimisation can be performed. Moreover, in the current approach, all the sampling points have equal importance. A learning method could be used to assess the importance of each sampling point over an extended period. This could lead to further optimisation of the sampling strategy. It could also allow optimised data gathering over long periods helping monitor time-varying phenomena.

8.3 Summary

The work presented in this thesis proposed and validated novel approaches for optimising inspection and sensing missions undertaken by a team of robots. These approaches target cases where there is a limitation in the team's communication capabilities, as well as, in the team's mission execution budget. The proposed

approaches can optimise missions in an online manner allowing the use in autonomous platforms. In addition, this work has identified potential extensions of it regarding the networking protocols it uses, its information communication strategy, its task allocation methods and the parameters of the sampling strategy that is used. These improvements have the potential to extend the autonomy to larger teams of robots performing persistent monitoring scenarios.

Appendices

Appendix A

On optimisation

This thesis studies the optimisation of the behaviour and performance of teams of robots with an application to underwater vehicles. This chapter acts as an introduction to optimisation, aiming to present to the reader useful methods and concepts that will be used in the following chapters. This will be done by presenting one of the most famous optimisation problem in the literature, as well as solution methods that were introduced over the years.

A.1 The travelling salesman problem

The *Travelling Salesman Problem* (TSP) is one of the classic optimisation problems presented in the literature. It has drawn much attention since the 1950s and was the main force leading the research in the field of *Operational Research* (OR) generating methods to solve multiple optimisation problems in the real world.

In this is problem, there is a single agent, called the salesman, who wants to visit a set of vertices. This agent can represent a vehicle, whose path needs to be optimised. The vertices that need to be visited are representing cities in the TSP or points for making sensor measurements in the vehicle case. Aim of the salesman is to visit all the cities following a path with the minimum travelling distance or cost.

The salesman is starting his route form a specific city and must finish his path in that city. This way a cyclic route is formed starting and ending on the same city. In the vehicle case that city is the starting point of the mission where the vehicle is deployed, and from where it will be recovered after the mission end. A sample solution for such a problem can be seen in figure A.1.

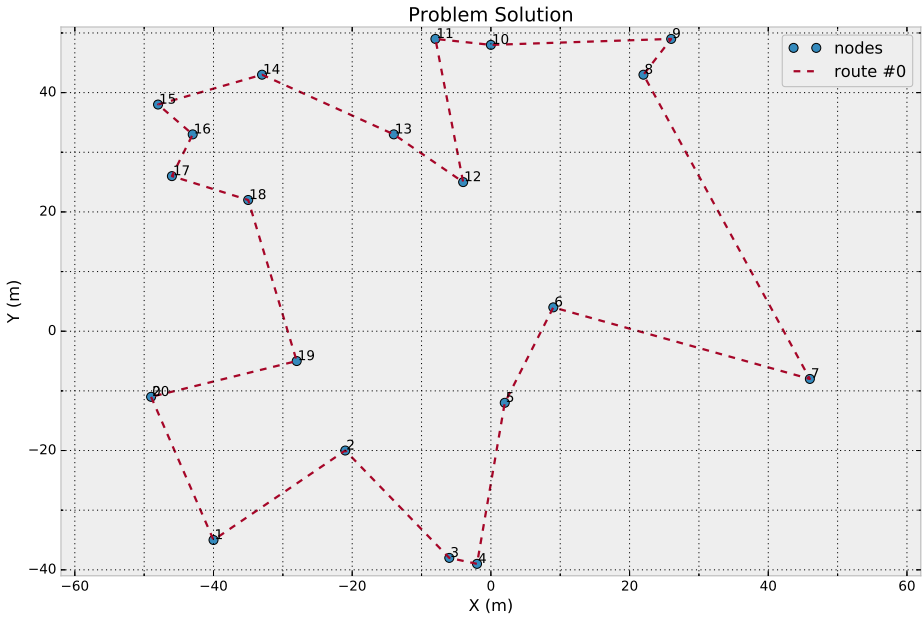


Figure A.1. Solution to a randomly generated TSP instance. 20 cities were randomly selected in an area of 100 by 100 units. The total cost of the route is 409.251 units. The route starts and ends at the leftmost city.

The solution to any TSP problem can be described as a minimisation problem satisfying a set of constraints. An *Integer Linear Programming* (ILP) formulation of the problem was first presented in [127] and is accordingly presented.

$$\min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} c_{ij} x_{ij} \tag{A.1}$$

$$s.t.\ 0 \leq x_{ij} \leq 1 \qquad i, j = 0, ..., n \tag{A.2}$$

$$u_i \in \mathbb{Z}_0^+ \qquad i = 0, ..., n \tag{A.3}$$

$$\sum_{i=0, i \neq j}^{n} x_{ij} = 1 \qquad j = 0, ..., n \tag{A.4}$$

$$\sum_{j=0, j \neq i}^{n} x_{ij} = 1 \qquad i = 0, ..., n \tag{A.5}$$

$$u_i - u_j + n x_{ij} \leq n - 1 \qquad 1 \leq i \neq j \leq n \tag{A.6}$$

The objective function that is minimised can be seen in (A.1). Variable x_{ij} is used to denote the existence of a path from vertex i to vertex j. Variable c_{ij}

represents the cost to travel between vertices i and j. In the general case of the TSP, the travelling cost is assumed to be symmetric, meaning that the cost to travel from vertex i to vertex j is the same as to travel from vertex j to vertex i. Over the years solutions have also appeared for the asymmetric TSP, for example in [128].

Constraint (A.2) sets the variable x_{ij} to be binary. The variable u_i, found in constraint (A.3), stores the order that each vertex is visited in the solution. Constraints (A.4) and (A.5) ensure that in each vertex one path enters and one path exits. Finally, constraint (A.6) enforces the presence of a single path traversing all the cities. These types of constraints are called sub-tour elimination constraints.

To solve the above minimisation problem, various solution methods have been introduced. Some of them aim at solving the problem exactly, obtaining the optimal solution, while some of them provide heuristic methods that can get close to optimal approximations. Both of the solution approaches will be presented in the following sections.

A.2 Exact solution methods

In the previous section, a famous trajectory optimisation method was presented, along with an ILP formulation of it. This section will provide the essential tools for finding the optimal solutions to this family of optimisation problems.

A.2.1 Linear programming

The integer linear programming formulations presented is a subset of the more general set of linear problems. In the integer linear programming, the problem variables are required to take integer values, while in the more generic linear programs are allowed to take any real value.

The first formulation of linear programming problems was presented by Dantzig [129]. He defined linear programs to be optimisation problems where the function to be optimised is linear and that they are constrained by linear inequalities. Danzig also presented the simplex method for linear optimisation. In this section, the simplex method will be briefly presented through an example found in [130].

Assume that there is the following LP to be solved (i.e. to have (A.7) max-

imised).

$$\max\ 5x_1 + 4x_2 + 3x_3 \tag{A.7}$$
$$s.t.\ 2x_1 + 3x_2 + x_3 \leq 5 \tag{A.8}$$
$$4x_1 + x_2 + 2x_3 \leq 11 \tag{A.9}$$
$$3x_1 + 4x_2 + 2x_3 \leq 8 \tag{A.10}$$
$$x_1, x_2, x_3 \geq 0 \tag{A.11}$$

The initial step is to transform all the inequalities to equalities adding slack variables. After that the problem is the following:

$$\max\ \zeta = 5x_1 + 4x_2 + 3x_3 \tag{A.12}$$
$$s.t.\ w_1 = 5 - 2x_1 - 3x_2 - x_3 \tag{A.13}$$
$$w_2 = 11 - 4x_1 - x_2 - 2x_3 \tag{A.14}$$
$$w_3 = 8 - 3x_1 - 4x_2 - 2x_3 \tag{A.15}$$
$$x_1, x_2, x_3, w_1, w_2, w_3 \geq 0 \tag{A.16}$$

Essence of the simplex method is to find an initial solution that complies with the constraints described in (A.13) - (A.16) and iteratively improve it until it maximises the objective function (A.12). The variables appearing on the left hand side in the constraints (A.13) - (A.15) are called the basic variables, forming the set $\mathcal{B}$. Similarly, the non-dependant variables are called non-basic variables and form the set $\mathcal{N}$. The whole set of equations (A.12) - (A.15) is referred as a *dictionary*.

An initial feasible solution to this problem is setting the original variables to zero. This would lead the objective function ζ to also have the value of 0 and the slack variables to become $w_1 = 5, w_2 = 11, w_3 = 8$.

To improve this solution one must look at the coefficients of the non-basic variables in the objective function. Since all are positive, it means that altering the value of each one of them will lead to an improvement to the objective value. In each iteration, one variable that improves the solution is chosen. This variable is named the *entering variable* as it leaves $\mathcal{N}$ and enters $\mathcal{B}$. In the same manner, a variable from $\mathcal{B}$ is selected that will enter $\mathcal{N}$. This is called the *leaving variable*.

This variable is selected based on preserving the non-negative status of all the other variables.

For this step, x_1 is chosen to be the entering variable. It can be seen that as x_1 will increase it may cause the basic variables to become negative. The maximum allowed increase must, therefore, be calculated. For w_1, given that x_2 and x_3 are kept to zero, one can calculate the maximum solving the equation $5-2x_1 = 0$. Thus, the maximum allowed increase for x_1 given w_1 is $5/2$. Likewise, the maximum allowed values based on w_2 and w_3 are calculated to be $11/4$ and $8/3$ respectively.

Given the results it is found that the maximum allowed increase is $5/2$ and is dictated by basic variable w_1. This is the basic variable that will become the leaving variable chosen in this step. To perform that equation (A.13) must be solved for x_1 producing the following equation:

$$x_1 = 2.5 - 0.5w_1 - 1.5x_2 - 0.5x_3 \tag{A.17}$$

Given equation (A.17), x_1 is replaced in all instances it appears as a non-basic variable leading to the following dictionary:

$$\zeta = 12.5 - 2.5w_1 - 3.5x_2 + 0.5x_3 \tag{A.18}$$

$$x_1 = 2.5 - 0.5w_1 - 1.5x_2 - 0.5x_3 \tag{A.19}$$

$$w_2 = 1 + 2w_1 + 5x_2 \tag{A.20}$$

$$w_3 = 0.5 + 1.5w_1 + 0.5x_2 - 0.5x_3 \tag{A.21}$$

That concludes the first iteration of the simplex algorithm. For the second iteration a new entering variable must be chosen. Looking at the objective function it is obvious that there is only one candidate not lowering its value. That is x_3 and is the next entering variable. Solving for the maximum increase one can find it is 5 and 1 based on (A.19) and (A.21) respectively.This makes w_3 the leaving variable. The dictionary produced is the following:

$$\zeta = 13 - w_1 - 3x_2 - w_3 \tag{A.22}$$

$$x_1 = 2 - 2w_1 - 2x_2 + w_3 \tag{A.23}$$

$$w_2 = 1 + 2w_1 + 5x_2 \tag{A.24}$$

$$x_3 = 1 + 3w_1 + x_2 - 2w_3 \tag{A.25}$$

Given that all the coefficients in (A.22) are negative, an entering variable cannot be chosen as it would decrease the objective function value. This happens because all the variables are constrained to be non-negative. This terminates the simplex algorithm finding the optimal value. It can be calculated by setting the independent variables to zero. This would give a value of $\zeta = 13$.

Geometrical interpretation of a LP

In cases where the variables of a linear programming problem are less than three, one can visualise the constraints along with the set of feasible solutions of the objective function. Using that plot one can find the optimal solution. The following example from [130] will show the geometric representation of a linear programming problem. Let (A.26) be the objective function to be maximised under constraints (A.27)-(A.30).

$$\max \ 3x_1 + 2x_2 \tag{A.26}$$

$$s.t. \ -x_1 + 3x_2 \leq 12 \tag{A.27}$$

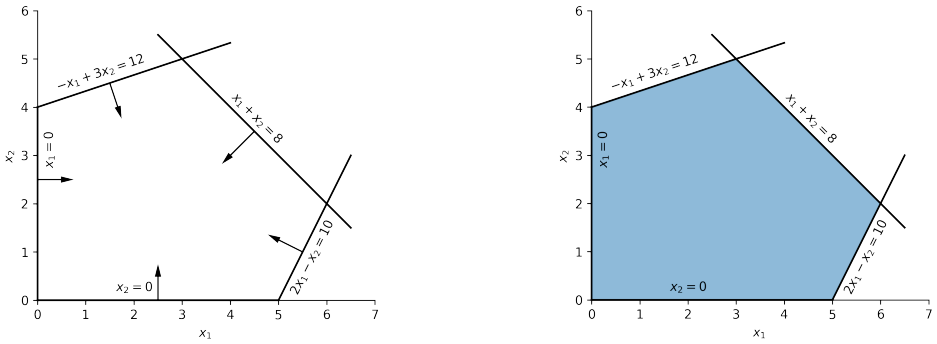
$$x_1 + x_2 \leq 8 \tag{A.28}$$

$$2x_1 - x_2 \leq 10 \tag{A.29}$$

$$x_1, \ x_2 \geq 0 \tag{A.30}$$

For each constraint present, a half-plane is formed. To plot these half planes, one can substitute the inequality with equality and plot the equivalent line. The half-plane can be calculated by selecting a point off the line and seeing if it satisfies the constraint or not. In figure A.2a one can see the constraints of the aforementioned problem. The half-plane that each constraint is defining is indicated by the direction of the arrow.

The area that is defined by the half-planes of the constraints is called the



(a) **(b)**

Figure A.2. Geometrical representation of constraints and feasible region for a linear programming problem. The half-plane formed by each constraint is indicated by a line and an accompanying arrow in A.2a. In A.2b the polygon represents the feasible region for the problem. Its' edges are formed by the according constraints.

feasible region of the problem. The optimal solution for the problem will be in one of the points of the feasible region. For the problem studied here the feasible region can be seen in figure A.2b.

The objective function can then be plotted on the same graph as the constraints. This can be seen in figure A.3. As the value of the objective function increases, the line that represents it moves to the right of the plot. The maximum can be found as a single point where the objective function touches the feasible region. For this example is at point (6,2) where the objective function takes the value 22. In general in an optimisation problem, the maximum (or minimum) will be found in one of the corners of the feasible region, formed by the constraints.

A.2.2 Branching

The simplex method, presented in the previous section, provides an excellent method for generic linear optimisation problems. However, when integer solutions are required, the simplex method may fail to find a solution.

To address that it was suggested to follow a divide and conquer strategy. In this strategy the requirement for an integer solution is initially dropped, thus forming a problem able to be solved by the simplex method. This is called an LP relaxation of an ILP. The relaxed problem is then solved to optimality. To move towards an integer solution, the optimal relaxation is then split into two

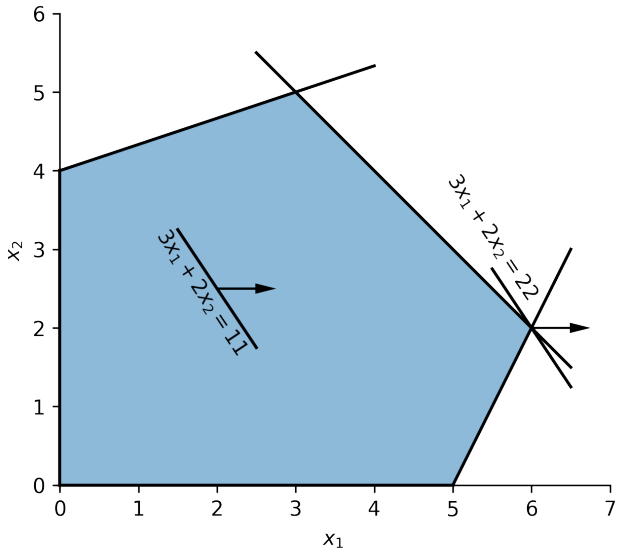


Figure A.3. Geometrical representation of the objective function of a linear programming problem. The function can take any value within the feasible region. The functions value increases as it moves from left to right. The optimal is found in one of the edges of the feasible region. Here the optimal has a value of 22.

different problems by choosing a violated integer variable having some real value and giving it the closest integer values. The two sub-problems can then have an LP relaxation computed and iteratively branched until a solution is found.

The suggested strategy was described in [131], [132] and is called branch and bound. The name incorporates the two main concepts in that method, branching as described before, and bounding. Bounding refers to the best bound on the optimal value that is calculated by the relaxed ILP. This value suggests that given those assumptions the optimal value of the integer problem will not be lower (or greater for a maximisation problem) than the one produced from the LP relaxation.

One can imagine this process as a tree having as its root the initial problem that was split and branches spanning out of it. This method has the benefit of searching the solution space and pruning regions where the value of the objective function is not improved. This is done by comparing the value of the objective function of each leaf LP relaxation on the tree with the best known valid solution. If the LP relaxation is worse than that solution, then any further search on that branch stops.

As with the generic LP, when the variables to be optimised are less than 3 the branching method can be visualised. Following an example from [133] will be presented and visualised to help the reader understand the methods mechanics. Let the following integer optimisation problem:

$$\max \ 5x_1 + 8x_2 \tag{A.31}$$

$$s.t. \ x_1 + x_2 \leq 6 \tag{A.32}$$

$$5x_1 + 9x_2 \leq 45 \tag{A.33}$$

$$x_1, x_2 \in \mathbb{Z}^{\geq 0} \tag{A.34}$$

In figure A.4 one can see the feasible region L_0 of the aforementioned problem. The dots in the feasible region represent feasible integer points. The dashed line shows the non-integer optimal solution, while the other lines represent the problem's constraints.

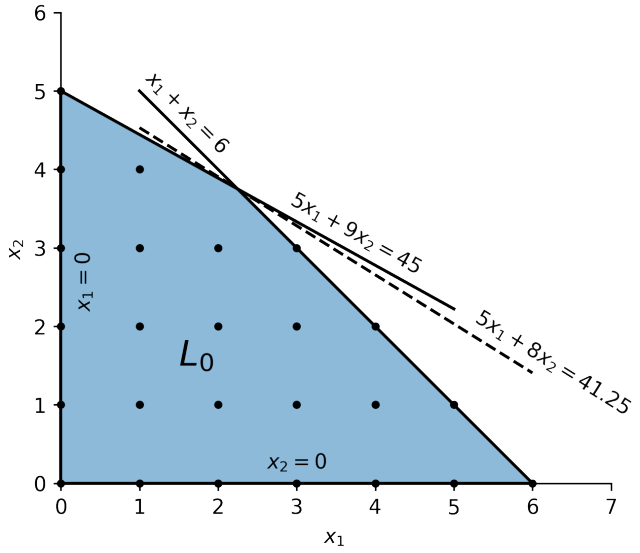


Figure A.4. Feasible region of an integer program. L_0 is the feasible region and the dots within it represent feasible integer points. The dashed line represents the non-integer optimal solution. The other lines represent the constraints.

The continuous optimal solution is found in (2.25, 3.75) with value 41.25. The closest integer solution to that can be obtained by rounding the optimal values to (2, 4). Unfortunately, this solution is not feasible given the problem's constraints.

Given the optimal value of the LP problem, one can form an upper bound for the optimal value of the ILP. It can be easily shown that the ILP optimal value $z^* \leq 41$.

Examining the LP solution one can start dividing the feasible region into smaller parts in order to satisfy the integer constraints of the problem variables. For example, the optimal value for x_2 in the LP is 3.75. In the equivalent ILP the value of x_2 should be integer giving two feasible regions for the cases of $x_2 \leq 3$ and $x_2 \geq 4$ as it can be seen in figure A.5.

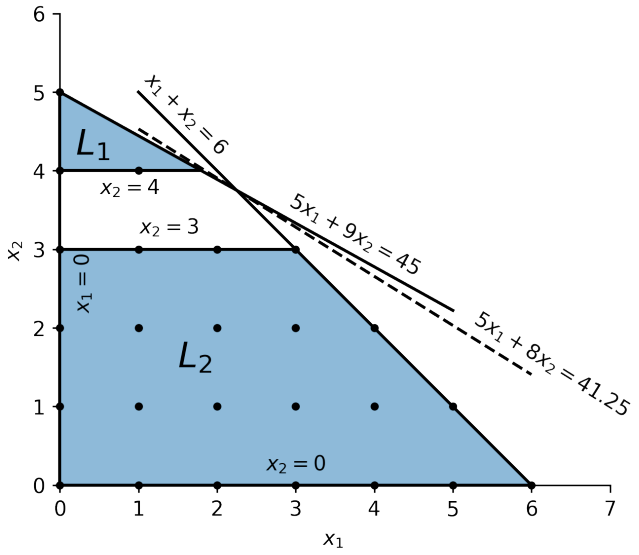


Figure A.5. Feasible regions of an integer program created after branching. L_1 is the feasible region for the case where $x_2 \geq 4$ while L_2 for the case where $x_2 \leq 3$.

The equivalent *enumeration tree* that is used to represent the branch and bound method can be seen in figure A.6. It can be seen that two branches were created, based on the additional constraints put on variable x_2. Each of the branches is representing a search direction in the two feasible regions created, L_1 and L_2 respectively.

The search will continue by expanding each of the two branches sequentially. The branch L_1 with $x_2 \geq 4$ will be expanded first. Taking a closer look in figure A.5, one can see that the optimal LP solution for L_1 is found for $x_2 = 4$, $x_1 = 1.8$ and $z = 41$. Given that x_1 is not integer branches are created for the cases of $x_1 \leq 1$ and $x_1 \geq 2$. These cases subdivide of L_1 to L_3 and L_4 respectively as can be seen in figure A.7. It should be noted that region L_3 is not visualised as the

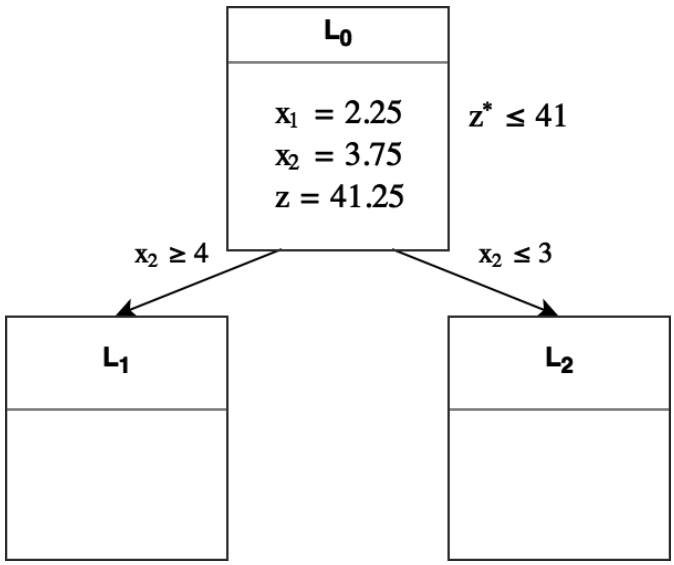


Figure A.6. Initial enumeration tree for the integer programming branch and bound solution.

solution for that branch is infeasible.

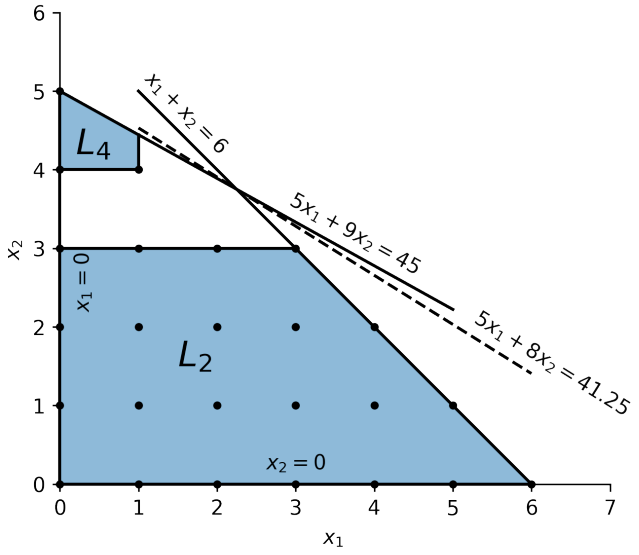


Figure A.7. Feasible regions after subdividing region L_1.

The enumeration tree after splitting branch L_1 to two sub-branches can be seen in figure A.8. The infeasible branch is marked with a $*$ symbol denoting that it will not be searched more.

The next step to the search procedure is to calculate the optimal LP values for L_4. From the geometric representation of figure A.8 it can be seen that $x_1 = 1$, $x_2 = 4.\overline{4}$ and $z = 40.\overline{5}$. Given that x_2 should be an integer L_4 must be divided

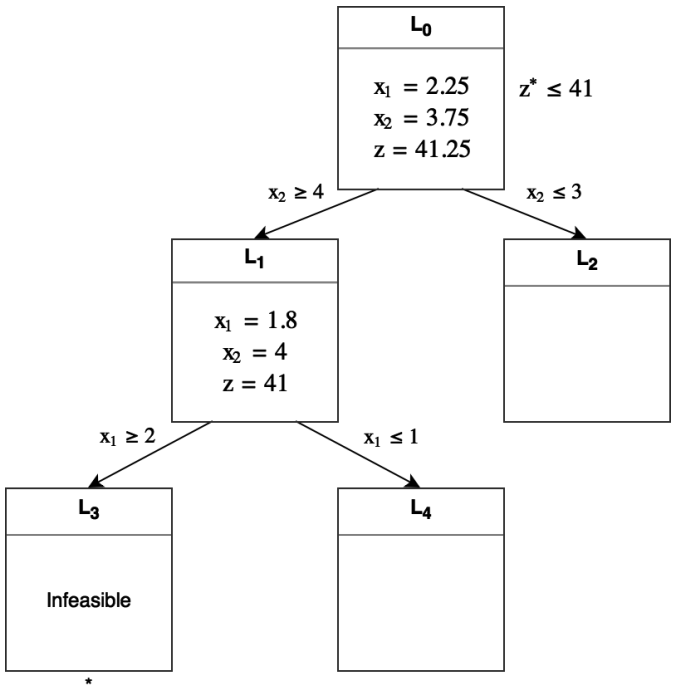


Figure A.8. Enumeration tree after splitting L_1 to L_3 and L_4. Since L_3 is infeasible it is marked with a $*$ and that branch will not be searched more.

into two regions L_5 and L_6 for the cases of $x_2 \leq 4$ and $x_2 \geq 5$ respectively. This division can be seen in figure A.9.

The search continues by exploring branches L_5 and L_6. Finding the optimal LP solution for L_5 gives the values $x_1 = 1$, $x_2 = 4$ and $z = 37$. This solution is also a feasible solution for the integer problem which has now gotten a lower bound of 37. Since the optimal solution for L_5 is an integer one, that branch will not be explored further. Branch L_6 has a single feasible point, namely $x_1 = 0$, $x_2 = 5$, which has an objective value of $z = 40$. This solution is updating the lower bound of the ILP making it $40 \leq z^* \leq 41$ and stopping the exploration on L_6. The exploration tree after these steps can be seen in figure A.10.

The only active branch after exploring L_5 and L_6 is L_2. Finding the optimal LP value for L_2 gives the values $x_1 = 3$, $x_2 = 3$ and $z = 39$. Since both x_1 and x_2 are integers, this solution is an acceptable solution for the ILP, and this branch will not be explored any more. The final enumeration tree can be seen in figure A.11. In case L_2 did not give an integer solution, exploration would still stop as the upper bound for this branch is less than the lower bound discovered by L_5.

After all branches have been explored, the branch that gave the highest value

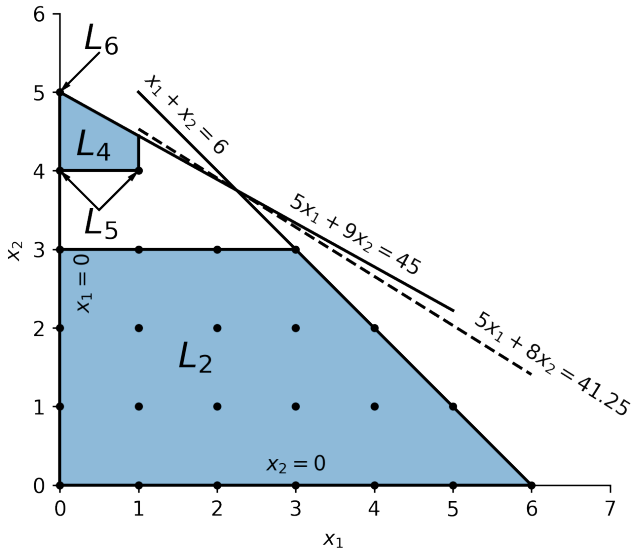


Figure A.9. Feasible regions after subdividing region L_4.

to the objective function is the one that has calculated the optimal value for the ILP. In this case the optimal value for z is 40 with $x_1 = 0$ and $x_2 = 5$. The ILP objective function along with the constraints and the feasible regions used for the branch and bound method can be seen in figure A.12.

A.2.3 Cutting planes

Another method that allows solutions to be found for an ILP was presented in [134]. This method is also based on initially solving an LP relaxation of the ILP. Then it iteratively adds constraints that transform the invalid LP relaxation to a valid ILP. The basic idea is that if for example the inequality $2x_1 + 3x_2 \leq 7.14$ is satisfied in the LP solution, then, the inequality $2x_1 + 3x_2 \leq 7$ will be valid for any integer solution. This inequality adds a cut to the solution space limiting the area that can be searched for a solution. The algorithm iteratively adds cuts and solves the problem until an optimal integer solution is found. Following an example will be presented. Let (A.35) be the objective function of an integer programming problem having constraints (A.36)-(A.39).

$$\max \ 4x_1 - x_2 \tag{A.35}$$

$$s.t. \ 7x_1 - 2x_2 \leq 14 \tag{A.36}$$

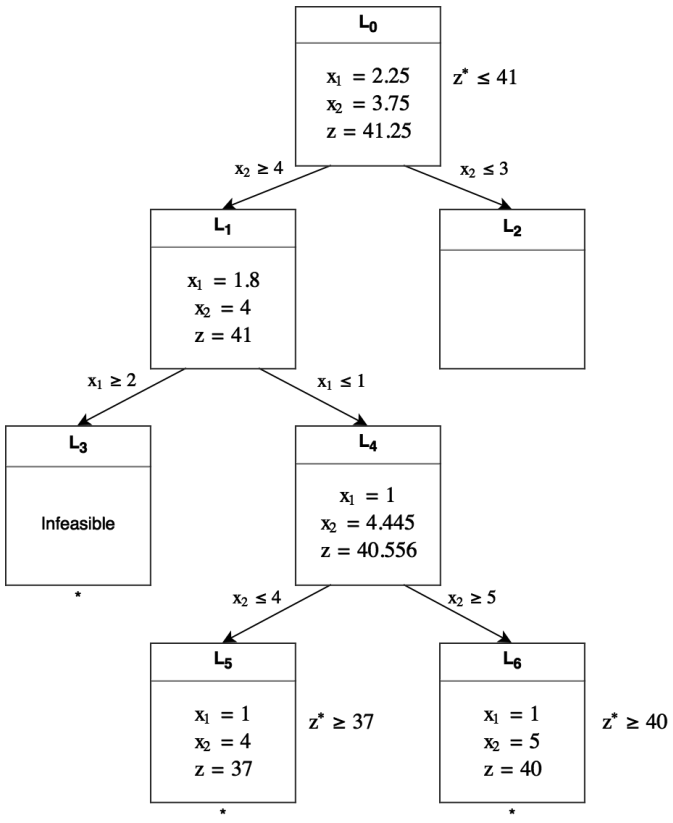


Figure A.10. Enumeration tree after exploring branches L_5 and L_6.

$$x_2 \leq 3 \tag{A.37}$$

$$2x_1 - 2x_2 \leq 3 \tag{A.38}$$

$$x_1, x_2 \in \mathbb{Z}^{\geq 0} \tag{A.39}$$

One can see the polyhedron described by the constraints, as well as the feasible points in figure A.13. Adding slack variables leads to the following LP.

$$\max \ 4x_1 - x_2 \tag{A.40}$$

$$s.t. \ 7x_1 - 2x_2 + x_3 = 14 \tag{A.41}$$

$$x_2 + x_4 = 3 \tag{A.42}$$

$$2x_1 - 2x_2 + x_5 = 3 \tag{A.43}$$

$$x_1, x_2, x_3, x_4, x_5 \in \mathbb{Z}^{\geq 0} \tag{A.44}$$

Solving the above LP relaxing the integer constraint using the simplex algo-

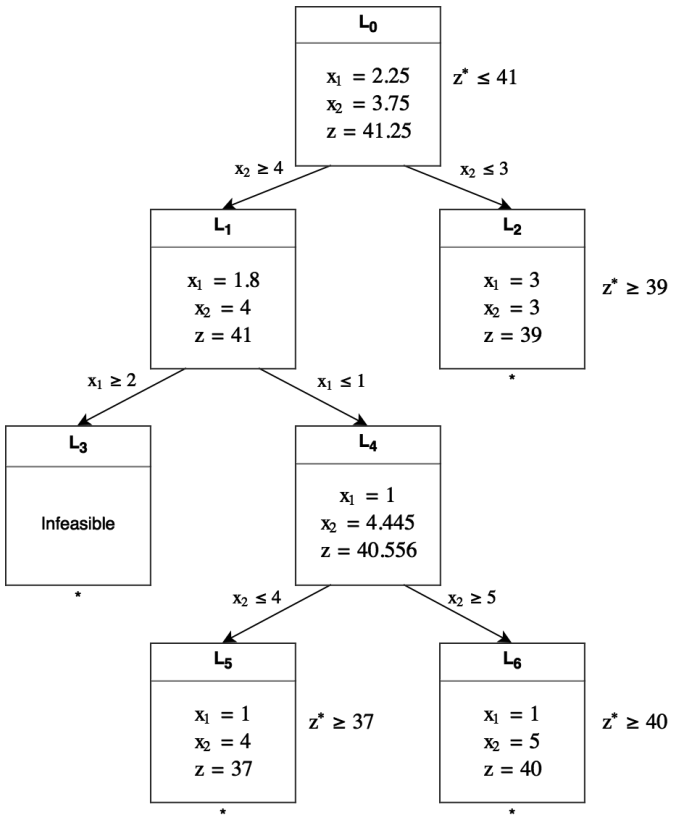


Figure A.11. Final enumeration tree after exploring all branches.

rithm as described before gives the following result.

$$\max\ 59/7 \qquad -4/7x_3 \quad -1/7x_4 \tag{A.45}$$

$$x_1 \quad +1/7x_3 \quad +2x_4 \quad =20/7 \tag{A.46}$$

$$x_2 \quad +x_4 \quad =3 \tag{A.47}$$

$$-2/7x_3 \quad +10/7x_4 \quad +x_5 \quad =23/7 \tag{A.48}$$

From (A.46) it can be seen that x_1=20/7. Given that it is fractional, a cut will be introduced according to the Gomory-Chvátal algorithm. The generated cut is shown in (A.50).

$$(\frac{1}{7} - \lfloor\frac{1}{7}\rfloor)x_3 + (\frac{2}{7} - \lfloor\frac{2}{7}\rfloor)x_4 \geq (\frac{20}{7} - \lfloor\frac{20}{7}\rfloor) \tag{A.49}$$

$$\Leftrightarrow \frac{1}{7}x_3 + \frac{2}{7}x_4 \geq \frac{6}{7} \tag{A.50}$$

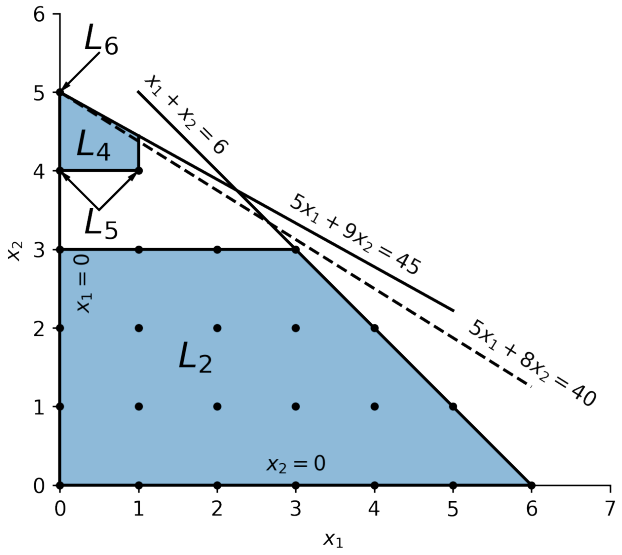


Figure A.12. Geometric representation of the objective function after finding the optimal solution. It can be seen that the integer optimal solution is on the top of the feasible region.

Given the cut of (A.50), the new constraint of (A.51) will be added to the original LP problem described by (A.40)-(A.44).

$$\frac{1}{7}x_3 + \frac{2}{7}x_4 + s = \frac{6}{7} \tag{A.51}$$

Given (A.41) and (A.42) one can calculate that $x_3 = 14 - 7x_1 + 2x_2$ and $x_4 = 3 - x_2$ respectively. Substituting them to (A.50) the cutting plane of (A.53) is obtained. It can be seen in figure A.14.

$$\frac{1}{7}(14 - 7x_1 + 2x_2) + \frac{2}{7}(3 - x_2) \geq \frac{6}{7} \tag{A.52}$$

$$\Leftrightarrow x_1 \leq 2 \tag{A.53}$$

Solving the original LP relaxation including the new constraint results to the following:

$$\max \ 15/2 - 1/5x_5 - 3s \tag{A.54}$$

$$x_1 + s = 2 \tag{A.55}$$

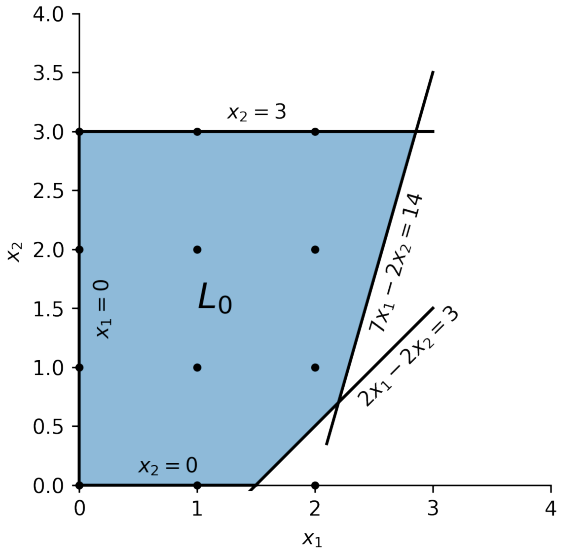


Figure A.13. Geometric representation of the polyhedron forming the integer programming problem. The dots show feasible integer solutions.

$$x_2 - 1/2s = 1/2 \tag{A.56}$$

$$x_3 - x_5 - s = 1 \tag{A.57}$$

$$x_4 + 1/2x_5 + 6s = 5/2 \tag{A.58}$$

One can see that x_2 gets the value of 1/2 from (A.56). Therefore a new cut will be generated for x_2. The new cut will be the one shown in (A.60).

$$(-\frac{1}{2} - \lfloor -\frac{1}{2} \rfloor)x_5 \geq (\frac{1}{2} - \lfloor \frac{1}{2} \rfloor) \tag{A.59}$$

$$\Leftrightarrow \frac{1}{2}x_5 \geq \frac{1}{2} \tag{A.60}$$

From (A.60) the following constraint (A.61) is derived and added to the original LP problem.

$$\frac{1}{2}x_5 - t = \frac{1}{2} \tag{A.61}$$

As done previously, the new constraint can be viewed in terms of the original variables of the LP, namely x_1 and x_2. This can be seen in (A.63) and figure A.15.

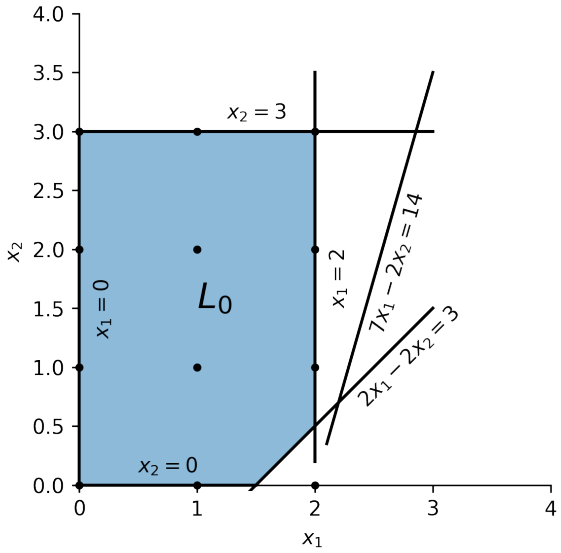


Figure A.14. Geometric representation of the polyhedron forming the integer programming problem after the first cut applied for $x_1 \leq 2$. The blue coloured area is the feasible region and the dots show feasible integer solutions.

$$\frac{1}{2}(2x_1 - 2x_2) - t = \frac{1}{2} \tag{A.62}$$

$$\Leftrightarrow x_1 - x_2 \leq 1 \tag{A.63}$$

Finally, solving again the original relaxation of the LP the following is obtained:

$$\max \; 7 - 3s - t \tag{A.64}$$

$$x_1 + s \quad = 2 \tag{A.65}$$

$$x_2 + s - t \; = 1 \tag{A.66}$$

$$x_3 - 5s - 2t = 2 \tag{A.67}$$

$$x_4 + 6s + t \; = 2 \tag{A.68}$$

$$x_5 - t = 1 \tag{A.69}$$

Since the optimal basic solution for the relaxed LP is integral, the solution for the integer problem is also integral and found at $(x_1, x_2) = (2, 1)$.

In [134] it was shown that if a sufficient number of cuts were added the algorithm always converges to an optimal integer solution. One problem is that the

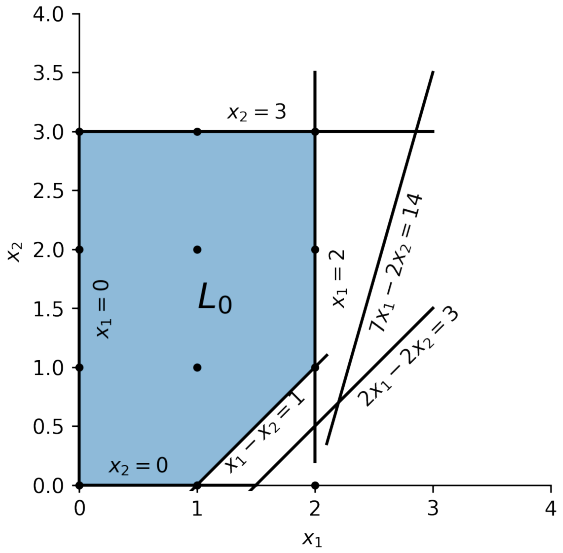


Figure A.15. Geometric representation of the polyhedron forming the integer programming problem after the second cut is applied for $x_1 - x_2 \leq 1$. The blue coloured area is the feasible region and the dots show feasible integer solutions.

number of constraints needing to be introduced to find the optimal solution can be significant. The algorithm may be wasting time adding cuts that improve the solution by an insignificant fraction that can lead to numerical instabilities.

In [135] it was proposed to combine the two aforementioned methods forming the branch and cut algorithm. This algorithm uses the cutting planes method to introduce cuts until no more useful cuts can be made. Then it branches and tries to solve the sub-problems using a branch and cut method. In this way, the numerical instabilities can be avoided. Branch and cut was found to be really efficient in finding solutions and is used in most commercial solvers today.

A.3 Heuristic solution methods

The exact solution methods presented in the previous sections have proven their usefulness in numerous cases. Unfortunately, the nature of the ILP and MIQP problems can make them hard to solve, requiring too much time and computational power to produce even a close to the optimal solution. For that reason, multiple heuristic techniques have been proposed over the years. These methods aim to find optimised solutions using search techniques which are computationally cheaper

than the exact methods and have demonstrated their real-world applicability in multiple occasions. This section will briefly present some of the main heuristics used in optimisation literature.

A.3.1 Ant colony optimisation

The *ant colony optimisation* (ACO) method was originally presented in [136]. It is a construction based search method. These methods try to find the optimal solution to the problem by iteratively constructing solutions that are improved using some statistical measure for choosing how the solution is constructed.

The ACO method is inspired by the way ants are gathering food in nature. Several species of ants are capable of marking their environment with a pheromone that other ants can detect. Using this strategy, each ant can mark the path it followed to reach a food source from the nest. Other ants sensing these pheromone trails can find the shortest path leading to the food source. This was experimentally shown in [137] and [138]. In these studies, ants were presented with two different paths reaching the food source that were different in length. It was observed that ants initially were exploring both paths to the food, but after some time all the ants were using the shortest path.

Explanation of that behaviour can be found in the way the ants deposit the aforementioned pheromone. At the beginning of the experiment, there is no pheromone on any of the two paths. This made the ants choose randomly which path to take. Given the length difference of the two paths, ants taking the shortest one are reaching the food faster and begin to transport it back to the nest. After depositing food back to the nest, there is a higher probability to choose the path with higher pheromone levels to travel back to the source of food. This way one of the two paths is gathering higher pheromone levels and most of the ants choose that optimising their travelling distance. A picture depicting the path followed in the experiment of [137] can be seen in figure A.16.

In the ACO method, multiple ants are used to produce different solutions. They produce artificial pheromone trails, which are used to indicate preference in the way a solution constructed. These trails are updated at run-time as more information is gathered regarding the problem solved. In the initial setting, where the artificial pheromone trails are not present, or at least not strong enough, more

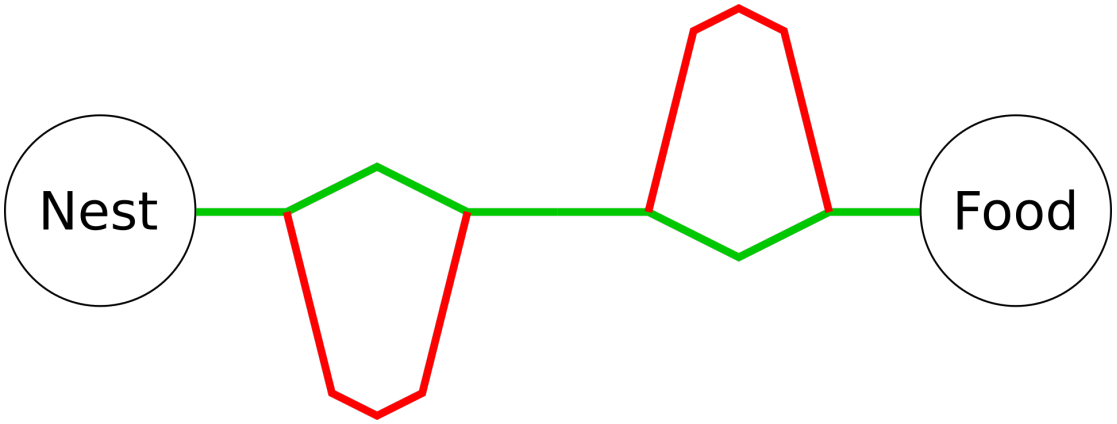


Figure A.16. Replication of the paths used in [137]. Ants learned to follow the shortest path to the food source (here depicted in green) using pheromone trails.

solutions are explored as the ants are randomly generating solutions. In the later stages, the solution space is much more limited, as solutions are built using prior experience, and the best solution candidates are generated.

In addition to the artificial pheromone trails, ACO methods apply various other techniques to improve the solution. For instance, each ant has memory that is allowing constraints to be implemented. Another technique allows the amount of pheromone that is deposited to be analogous to the quality of the produced solution. Pheromone evaporation assumes that the amount of pheromone decreases naturally over time if no action is taken. This helps the ACO to avoid getting stuck to local optima too early in the solution process. Daemon actions are methods applied to the whole system by an external force and are used to influence the system functionality. Each ant cannot generate any daemon action. For example, changing the pheromone concentration can lead the search strategy to particular solutions. An algorithmic view of the ACO can be seen in algorithm A.1.

More information on the topic can be found in [139] and [140] where the algorithm and application examples are explained in detail.

A.3.2 Tabu Search

Another type of heuristic methods are the ones that try to find the optimal solution through iterative modifications of a candidate solution. These modifications aim to change the candidate solution enough so that it reaches an optimal status. An exemplar method of this category is called *Tabu Search* (TS) and was presented

Algorithm A.1 Ant Colony Optimisation

Input: Number of ants n_a
Output: Best found solution s^*

```
1: InitialisePheromoneTrails
2: while termination criterion not satisfied do
3:     for k = 1 to n_a do
4:         s_k = ConstructSolution
5:         UpdatePheromoneTrails
6:     ApplyPheromoneEvaporation
7:     ApplyDaemonActions                          ▷ optional
8:     s* = UpdateBestFoundSolution
9: return s*
```

in [141]. An algorithmic representation of TS can be seen in algorithm A.2.

Algorithm A.2 Tabu Search

Input: Initial solution s
Output: Best found solution s^*

```
1: T = InitialiseTabuList
2: while termination criterion not satisfied do
3:     N = GetNeighbourhood(s)
4:     s = ChooseBestNonTabuSolution(N, T)
5:     T = UpdateTabuList(s)
6:     s* = UpdateBestFoundSolution(s)
7: return s*
```

The initial step is to initialise the tabu list. Then the algorithm iteratively tries to modify the solution until a termination criterion is met. Firstly, it gets the neighbourhood of candidate solution s, by performing all the possible modifications in that solution. Then the best allowable solution is chosen. This is performed by choosing the best neighbour that is reachable with a modification not in the tabu list. After the best neighbouring solution is chosen, the tabu list is updated with the modification that led to that solution. Finally, the best-found solution is updated with the current solution if that is applicable. If the termination criterion is reached, the iterative improvement process stops and the best-found solution to that time is returned.

One of the main concepts of the TS method, as revealed by its' name, is the tabu list. This list acts as a limited memory disallowing solutions to be revisited. This prevents the search to repeatedly visit the same solutions in a cyclic manner.

Additionally, the tabu list allows the TS method to avoid local optima. This is achieved by allowing the search to reach the local optimum, but then forcing it to take steps that reduce the solution quality. Taking enough steps will allow to escape the local optimum and reach better solutions. It must be noted that the search is prevented from reaching the local optimum again, and thus get stuck to it, as it prevents cycles in moves. An example of this behaviour can be seen in figure A.17.

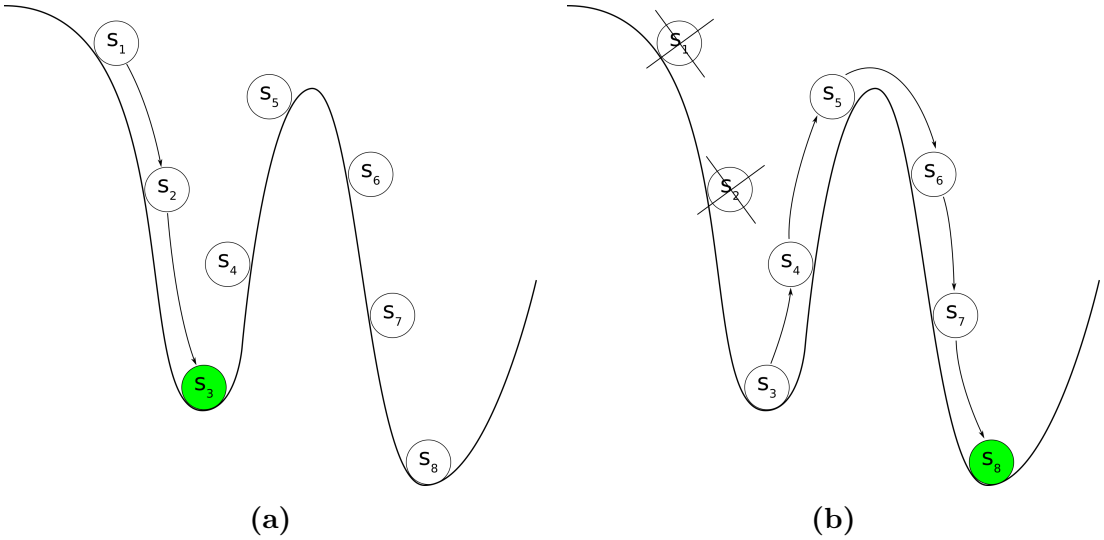


Figure A.17. Example of tabu search avoiding a local minimum. After solutions S_1 and S_2 the search arrives in local minimum S_3 as shown in (a). The next move is forcing the search to solution S_4 as transitions to the previous solutions are marked as tabu (crossed out in the figure). In the long run and by marking all previous solutions as tabu the search arrives in the improved minimum S_8 as it can be seen in (b).

A.3.3 Genetic Algorithms

The third type of heuristic is based on optimisation based on solution recombination. Solution recombination is based on mixing two solutions in search of a better solution. *Genetic Algorithms* (GA) is a prominent member of that family. They trace their routes in the research of adaptive systems during the 1960's [142], [143]. The first theoretical formulation was done by Holland in 1975 [144]. Since then they have been used in numerous applications. For example in [87], a genetic algorithm is used to optimise marine sampling strategies.

The theory behind GA is treating the optimisation problem the way evolution is driven by natural selection in the real world. In a GA there is a set of solutions, called the *population*. Every single solution in the population is called a *chromosome*. In accordance with the biological counterpart, chromosomes are constructed using *genes* whose specific values are called *alleles*. The optimisation process of a GA is described in algorithm A.3 [145].

Algorithm A.3 Genetic Algorithm

Output: *FittestChromosome*

1: InitialisePopulation
2: **while** *generation* $\leq$ *MaxGeneration* or *Convergence* **do**
3: Crossover
4: Mutate
5: SelectNewPopulation
6: Select *FittestChromosome*
7: **return** *FittestChromosome*

The initial step of a GA is to generate the population. This is performed either randomly or using some heuristic, and it aims to have a variation in the solutions produced. This will help to cover much of the search space and for characteristics belonging to good solutions to appear as genes. Then the iterative part of optimisation is performed until the population converges to some characteristic or the maximum amount of *generations* (i.e. iterations) is reached. In the iterative part of the algorithm consists of the functions of *crossover*, *mutation* and *selection* as would happen to a natural ecosystem.

Crossover is the action that recombines two solutions in the search for better solutions. It is the equivalent action of mating; therefore, the two chromosomes taking part are called *parents* and the solutions produced are called *offsprings*. In general, it is hoped that the recombination of good parts of the parents will produce better quality offsprings. A sample crossover operation can be seen in figure A.18.

Mutation is the process of introducing some randomness into the solution. As in the real world, mutations can alter the value of one or more genes in a chromosome. Mutations usually happen to a percentage of the population based on some problem defined *mutation rate*. A mutation example is depicted in figure A.19.

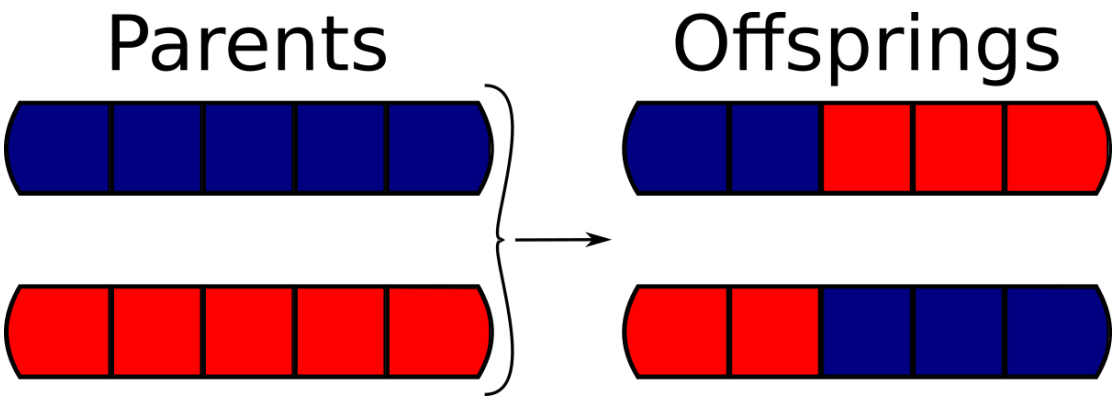


Figure A.18. A sample crossover operation. Parts of two parents are combined to create two offsprings.

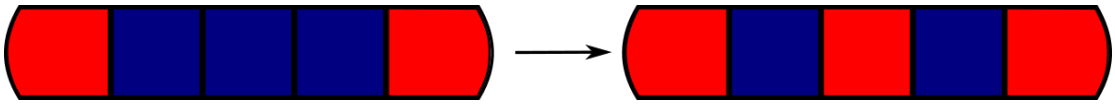


Figure A.19. A sample mutation operation. The third gene of the chromosome represented changes value to a red allele.

At the end of each iteration, a new population is selected that will be the next generation in the optimisation cycle. This happens by first evaluating each chromosome in the population. Evaluation is based on the objective function that needs to be optimised. Then the fittest chromosomes are selected and continue to the next generation. In the final step of the algorithm, the fittest chromosome is selected and is returned.

A.4 Summary

This appendix presented the required theory for solving optimisation problems. One of the most famous problems in the optimisation literature was presented. Along with that, methods to solve such problems exactly and approximately were detailed.

Bibliography

[1] W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff, "Autonomous and autonomic systems: A paradigm for future space exploration missions", *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 3, pp. 279–291, 2006.

[2] B. Allotta, R. Costanzi, A. Ridolfi, C. Colombo, F. Bellavia, M. Fanfani, F. Pazzaglia, O. Salvetti, D. Moroni, M. A. Pascali, *et al.*, "The arrows project: Adapting and developing robotics technologies for underwater archaeology", *IFAC-PapersOnLine*, vol. 48, no. 2, pp. 194–199, 2015.

[3] T. Tomic, K. Schmid, P. Lutz, A. Domel, M. Kassecker, E. Mair, I. L. Grixa, F. Ruess, M. Suppa, and D. Burschka, "Toward a fully autonomous uav: Research platform for indoor and outdoor urban search and rescue", *IEEE robotics & automation magazine*, vol. 19, no. 3, pp. 46–56, 2012.

[4] J. Bellingham and J. Willcox, "Optimizing auv oceanographic surveys", in *Autonomous Underwater Vehicle Technology, 1996. AUV '96., Proceedings of the 1996 Symposium on*, Jun. 1996, pp. 391–398. DOI: 10.1109/AUV.1996.532439.

[5] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems", *IJRR*, vol. 23, no. 9, 2004.

[6] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art", in *Cooperative Robots and Sensor Networks 2015*, Springer, 2015, pp. 31–51.

[7] M. Roth, D. Vail, and M. Veloso, "A real-time world model for multi-robot teams with high-latency communication", in *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International*

Conference on, vol. 3, Oct. 2003, 2494–2499 vol.3. DOI: 10.1109/IROS.2003.1249244.

[8] I. Vasilescu, K. Kotay, D. Rus, M. Dunbabin, and P. Corke, "Data collection, storage, and retrieval with an underwater sensor network", in *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems*, ser. SenSys '05, San Diego, California, USA: ACM, 2005, pp. 154–165, ISBN: 1-59593-054-X. DOI: 10.1145/1098918.1098936. [Online]. Available: http://doi.acm.org/10.1145/1098918.1098936.

[9] Z. Saigol, G. Frost, N. Tsiogkas, F. Maurelli, D. Lane, A. Bourque, and B. Nguyen, "Facilitating cooperative AUV missions: Experimental results with an acoustic knowledge-sharing framework", in *Proceedings of IEEE-MTS Oceans'13, San Diego, USA*, 2013.

[10] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to informative path planning for persistent monitoring tasks", in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, IEEE, 2014, pp. 342–349.

[11] J. Yu, M. Schwager, and D. Rus, "Correlated orienteering problem and its application to persistent monitoring tasks", *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1106–1118, Oct. 2016, ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2593450.

[12] F. Maurelli, Z. Saigol, J. Cartwright, D. Lane, A. Bourque, and B. Nguyen, "Tdma-based exchange policies for multi-robot communication of world information", in *IFAC MCMC*, 2012.

[13] I. Gurobi Optimization, *Gurobi optimizer reference manual*, 2014. [Online]. Available: http://www.gurobi.com.

[14] M. Brambilla, E. Ferrante, M. Birattari, and M. Dorigo, "Swarm robotics: A review from the swarm engineering perspective", *Swarm Intelligence*, 2013.

[15] L. E. Parker, "Alliance: An architecture for fault tolerant multirobot cooperation", *Robotics and Automation, IEEE Transactions on*, vol. 14, no. 2, 1998.

[16] R. Zlot and A. Stentz, "Market-based multirobot coordination for complex tasks", *The International Journal of Robotics Research*, vol. 25, no. 1, Jan. 2006, ISSN: 0278-3649. DOI: 10.1177/0278364906061160. [Online]. Available: http://ijr.sagepub.com/cgi/doi/10.1177/0278364906061160.

[17] G. A. Korsah, A. Stentz, and M. B. Dias, "A comprehensive taxonomy for multi-robot task allocation", *IJRR*, vol. 32, no. 12, 2013.

[18] H. M. La, W. Sheng, and J. Chen, "Cooperative and active sensing in mobile sensor networks for scalar field mapping", *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 1, pp. 1–12, 2015.

[19] J. Binney and G. S. Sukhatme, "Branch and bound for informative path planning", in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, IEEE, 2012, pp. 2147–2154.

[20] V. D. Jr. and O. A., "The personnel assignment problem", in *Symposium on Linear Inequalities and Programming*, 1952, pp. 155–163.

[21] H. W. Kuhn, "The hungarian method for the assignment problem", *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[22] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, M. Moors, S. Thrun, and H. Younes, "Coordination for multi-robot exploration and mapping", in *Aaai/Iaai*, 2000, pp. 852–858.

[23] B. Gerkey and M. Mataric, "Sold!: Auction methods for multirobot coordination", *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, Oct. 2002, ISSN: 1042-296X. DOI: 10.1109/TRA.2002.803462. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1067996.

[24] D. Vail, "Dynamic multi-robot coordination", *Multi-Robot Systems: From Swarms to intelligent automata*, 2003. [Online]. Available: http://books.google.com/books?hl=en%5C&lr=%5C&id=2H9pFRTNbRUC%5C&oi=fnd%5C&pg=PA87%5C&dq=DYNAMIC+MULTI-ROBOT+COORDINATION%5C&ots=L-0-lDpHnM%5C&sig=R41aP-cMDyitO8bB11doL3Lkr4Y%20http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=5152765%20http://books.google.com/books?hl=en%5C&lr=%5C&id=2H9pFRTNbRUC%5C&oi=fnd%5C&

pg=PA87%5C&dq=Dynamic+multi-robot+coordination%5C&ots=L-0-lDpHvF%5C&sig=QyvU3q4r2nJa2Ml8TxrfoBq-vFw.

[25] D. B. Shmoys and É. Tardos, "An approximation algorithm for the generalized assignment problem", *Mathematical programming*, vol. 62, no. 1-3, pp. 461–474, 1993.

[26] M. Savelsbergh, "A branch-and-price algorithm for the generalized assignment problem", *Operations research*, vol. 45, no. 6, pp. 831–841, 1997.

[27] P. Brucker, *Scheduling Algorithms (5th ed.)* Springer, 2006.

[28] P. Toth and D. Vigo, *The vehicle routing problem*. Society for Industrial and Applied Mathematics, 2001.

[29] T. Bektas, "The multiple traveling salesman problem: An overview of formulations and solution procedures", *Omega*, vol. 34, no. 3, 2006, ISSN: 0305-0483. DOI: http://dx.doi.org/10.1016/j.omega.2004.10.004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0305048304001550.

[30] B. L. Brumitt and A. Stentz, "Grammps: A generalized mission planner for multiple mobile robots in unstructured environments", in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, IEEE, vol. 2, 1998, pp. 1564–1571.

[31] J. Melvin, P. Keskinocak, S. Koenig, C. Tovey, and B. Y. Ozka, "Multi-robot routing with rewards and disjoint time windows", in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2007, pp. 2332–2337.

[32] M. B. Dias, "Traderbots : A new paradigm for robust and efficient multi-robot coordination in dynamic environments", PhD thesis, Carnegie Mellon University, 2004.

[33] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions", in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, IEEE, vol. 2, 2003, pp. 1957–1962.

[34] S. Koenig, C. A. Tovey, X. Zheng, and I. Sungur, "Sequential bundle-bid single-sale auction algorithms for decentralized control.", in *IJCAI*, 2007, pp. 1359–1365.

[35] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing", in *Robotics: Science and Systems*, Jun. 2005.

[36] J. B. Mazzola and A. W. Neebe, "Resource-constrained assignment scheduling", *Operations Research*, vol. 34, no. 4, pp. 560–572, 1986.

[37] D. Bredström and M. Rönnqvist, "Combined vehicle routing and scheduling with temporal precedence and synchronization constraints", *European journal of operational research*, vol. 191, no. 1, pp. 19–31, 2008.

[38] A. Dohn, M. S. Rasmussen, and J. Larsen, "The vehicle routing problem with time windows and temporal dependencies", *Networks*, vol. 58, no. 4, pp. 273–289, 2011.

[39] M. S. Rasmussen, T. Justesen, A. Dohn, and J. Larsen, "The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies", *European Journal of Operational Research*, vol. 219, no. 3, pp. 598–610, 2012.

[40] S. Botelho, "M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement", in *Proceedings of the 1999 IEEE International Conference on Robotics and Automation (ICRA99)*, 1999. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs%5C_all.jsp?arnumber=772530.

[41] D. C. MacKenzie, "Collaborative tasking of tightly constrained multi-robot missions", in *Multi-Robot Systems: From Swarms to Intelligent Automata: Proceedings of the 2003 International Workshop on Multi-Robot Systems*, Citeseer, vol. 2, 2003, pp. 39–50.

[42] S. Chien, A. Barrett, and T. Estlin, "A comparison of coordinated planning methods for cooperating rovers", in *4th International Conference on Autonomous Agents*, 2000. [Online]. Available: http://dl.acm.org/citation.cfm?id=337057.

[43] T. Lemaire, R. Alami, and S. Lacroix, "A distributed tasks allocation scheme in multi-uav context", in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, IEEE, vol. 4, 2004, pp. 3622–3627.

[44] A. R. Mosteo, L. Montano, and M. G. Lagoudakis, "Multi-robot routing under limited communication range", in *2008 IEEE International Conference on Robotics and Automation*, IEEE, 2008, pp. 1531–1536.

[45] E. Balas and M. W. Padberg, "Set partitioning: A survey", *SIAM review*, vol. 18, no. 4, pp. 710–760, 1976.

[46] S. D. Ramchurn, M. Polukarov, A. Farinelli, C. Truong, and N. R. Jennings, "Coalition formation with spatial and temporal constraints", in *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 3-Volume 3*, International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 1181–1188.

[47] O. Shehory and S. Kraus, "Task allocation via coalition formation among autonomous agents", in *IJCAI (1)*, Citeseer, 1995, pp. 655–661.

[48] ——, "Methods for task allocation via agent coalition formation", *Artificial intelligence*, vol. 101, no. 1-2, pp. 165–200, 1998.

[49] L. Vig and J. A. Adams, "Multi-robot coalition formation", *IEEE transactions on robotics*, vol. 22, no. 4, pp. 637–649, 2006.

[50] J. Guerrero and G. Oliver, "Multi-robot task allocation strategies using auction-like mechanisms", *Artificial Research and Development in Frontiers in Artificial Intelligence and Applications*, vol. 100, pp. 111–122, 2003.

[51] L. Lin and Z. Zheng, "Combinatorial bids based multi-robot task allocation method", in *Proceedings of the 2005 IEEE international conference on robotics and automation*, IEEE, 2005, pp. 1145–1150.

[52] P. M. Shiroma and M. F. Campos, "Comutar: A framework for multi-robot coordination and task allocation", in *2009 IEEE/RSJ international conference on intelligent robots and systems*, IEEE, 2009, pp. 4817–4824.

[53] M. Koes, I. Nourbakhsh, and K. Sycara, "Constraint optimization coordination architecture for search and rescue robotics", in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, IEEE, 2006, pp. 3977–3982.

[54] G. A. Korsah, B. Kannan, B. Browning, A. Stentz, and M. B. Dias, "Xbots: An approach to generating and executing optimal multi-robot plans with cross-schedule dependencies", in *2012 IEEE International Conference on Robotics and Automation*, IEEE, 2012, pp. 115–122.

[55] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs", *Operations research*, vol. 46, no. 3, pp. 316–329, 1998.

[56] E. G. Jones, M. B. Dias, and A. Stentz, "Time-extended multi-robot coordination for domains with intra-path constraints", *Autonomous robots*, vol. 30, no. 1, pp. 41–56, 2011.

[57] L. E. Parker and F. Tang, "Building multirobot coalitions through automated task solution synthesis", *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1289–1305, 2006.

[58] T. Tsiligirides, "Heuristic methods applied to orienteering", *Journal of the Operational Research Society*, vol. 35, no. 9, pp. 797–809, 1984.

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

[60] G. Laporte and S. Martello, "The selective travelling salesman problem", *Discrete applied mathematics*, vol. 26, no. 2-3, pp. 193–207, 1990.

[61] M. Gendreau, G. Laporte, and F. Semet, "A branch-and-cut algorithm for the undirected selective traveling salesman problem", *Networks*, vol. 32, no. 4, pp. 263–273, 1998.

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

[63] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications", *European Journal of Operational Research*, vol. 255, no. 2, pp. 315–332, 2016.

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

[65] M. Gendreau, G. Laporte, and F. Semet, "A tabu search heuristic for the undirected selective travelling salesman problem", *European Journal of Operational Research*, vol. 106, no. 2, pp. 539–545, 1998.

[66] Y.-C. Liang, S. Kulturel-Konak, and A. E. Smith, "Meta heuristics for the orienteering problem", in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, IEEE, vol. 1, 2002, pp. 384–389.

[67] M. F. Tasgetiren, "A genetic algorithm with an adaptive penalty function for the orienteering problem", *Journal of Economic and Social Research*, vol. 4, no. 2, pp. 1–26, 2001.

[68] M. Schilde, K. F. Doerner, R. F. Hartl, and G. Kiechle, "Metaheuristics for the bi-objective orienteering problem", *Swarm Intelligence*, vol. 3, no. 3, pp. 179–201, 2009.

[69] X. Wang, B. L. Golden, and E. A. Wasil, "Using a genetic algorithm to solve the generalized orienteering problem", in *The vehicle routing problem: Latest advances and new challenges*, Springer, 2008, pp. 263–274.

[70] J. Karbowska-Chilinska, J. Koszelew, K. Ostrowski, and P. Zabielski, "Genetic algorithm solving orienteering problem in large networks.", in *KES*, 2012, pp. 28–38.

[71] Z. Sevkli and F. E. Sevilgen, "Discrete particle swarm optimization for the orienteering problem", in *IEEE Congress on Evolutionary Computation*, IEEE, 2010, pp. 1–8.

[72] V. Campos, R. Martí, J. Sánchez-Oro, and A. Duarte, "Grasp with path relinking for the orienteering problem", *Journal of the Operational Research Society*, vol. 65, no. 12, pp. 1800–1813, 2014.

[73] Y. Marinakis, M. Politis, M. Marinaki, and N. Matsatsinis, "A memetic-grasp algorithm for the solution of the orienteering problem", in *Modelling, Computation and Optimization in Information Systems and Management Sciences*, Springer, 2015, pp. 105–116.

[74] I.-M. Chao, B. L. Golden, and E. A. Wasil, "The team orienteering problem", *European Journal of Operational Research*, vol. 88, no. 3, pp. 464–474, 1996, ISSN: 0377-2217. DOI: https://doi.org/10.1016/0377-2217(94)00289-4. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0377221794002894.

[75] H. Tang and E. Miller-Hooks, "A tabu search heuristic for the team orienteering problem", *Computers & Operations Research*, vol. 32, no. 6, pp. 1379–1407, 2005.

[76] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. Van Oudheusden, "A guided local search metaheuristic for the team orienteering problem", *European journal of operational research*, vol. 196, no. 1, pp. 118–127, 2009.

[77] ——, "Iterated local search for the team orienteering problem with time windows", *Computers & Operations Research*, vol. 36, no. 12, pp. 3281–3290, 2009.

[78] L. Ke, C. Archetti, and Z. Feng, "Ants can solve the team orienteering problem", *Computers & Industrial Engineering*, vol. 54, no. 3, pp. 648–665, 2008.

[79] N. Labadie, R. Mansini, J. Melechovskỳ, and R. W. Calvo, "The team orienteering problem with time windows: An lp-based granular variable neighborhood search", *European Journal of Operational Research*, vol. 220, no. 1, pp. 15–27, 2012.

[80] B. Aghezzaf and H. E. Fahim, "The multi-constraint team orienteering problem with time windows in the context of distribution problems: A variable neighborhood search algorithm", in *Logistics and Operations Management (GOL), 2014 International Conference on*, IEEE, 2014, pp. 155–160.

[81] S.-W. Lin and F. Y. Vincent, "A simulated annealing heuristic for the team orienteering problem with time windows", *European Journal of Operational Research*, vol. 217, no. 1, pp. 94–107, 2012.

[82] W. Souffriau, P. Vansteenwegen, G. V. Berghe, and D. Van Oudheusden, "A path relinking approach for the team orienteering problem", *Computers & operations research*, vol. 37, no. 11, pp. 1853–1859, 2010.

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

[84] C. Chekuri and M. Pal, "A recursive greedy algorithm for walks in directed graphs", in *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, IEEE, 2005, pp. 245–253.

[85] J. Binney, A. Krause, and G. S. Sukhatme, "Informative path planning for an autonomous underwater vehicle", in *Robotics and automation (icra), 2010 IEEE international conference on*, IEEE, 2010, pp. 4791–4796.

[86] L. Heng, A. Gotovos, A. Krause, and M. Pollefeys, "Efficient visual exploration and coverage with a micro aerial vehicle in unknown environments", in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, IEEE, 2015, pp. 1071–1078.

[87] S. Frolov, B. Garau, and J. Bellingham, "Can we do better than the grid survey: Optimal synoptic surveys in presence of variable uncertainty and decorrelation scales", *Journal of Geophysical Research: Oceans*, vol. 119, no. 8, pp. 5071–5090, 2014.

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

[89] A. Blum, S. Chawla, D. R. Karger, T. Lane, A. Meyerson, and M. Minkoff, "Approximation algorithms for orienteering and discounted-reward tsp", *SIAM Journal on Computing*, vol. 37, no. 2, pp. 653–670, 2007.

[90] R. Pěnička, J. Faigl, P. Váňa, and M. Saska, "Dubins orienteering problem", *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1210–1217, 2017.

[91] ——, "Dubins orienteering problem with neighborhoods", in *Unmanned Aircraft Systems (ICUAS), 2017 International Conference on*, IEEE, 2017, pp. 1555–1562.

[92] S. Arora and S. Scherer, "Randomized algorithm for informative path planning with budget constraints", in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, IEEE, 2017, pp. 4997–5004.

[93] C. J. Matheus, M. M. Kokar, and K. Baclawski, "A core ontology for situation awareness", in *Proceedings of the Sixth International Conference of Information Fusion, 2003*, vol. 1, 2003. DOI: 10.1109/ICIF.2003.177494. [Online]. Available: http://ftp.isif.org/fusion/proceedings/fusion03CD/special/s36.pdf.

[94] G. Papadimitriou and D. Lane, "Semantic based knowledge representation and adaptive mission planning for MCM missions using AUVs", in *IEEE-MTS Oceans'14, Taipei, Republic of China*, 2014.

[95] A. Jena, *A free and open source java framework for building semantic web and linked data applications*, 2017. [Online]. Available: https://jena.apache.org.

[96] P. Hintjens, *ZeroMQ: Messaging for many applications*. " O'Reilly Media, Inc.", 2013.

[97] E. GMBH, *Evologics s2cd 18/34*, 2017. [Online]. Available: https://www.evologics.de/en/products/acoustics/s2cr_18_34.html.

[98] N. S. Board, N. R. Council, *et al.*, *Naval mine warfare: Operational and technical challenges for naval forces*. National Academies Press, 2001.

[99] L. Freitag, M. Grund, C. Von Alt, R. Stokey, and T. Austin, "A shallow water acoustic network for mine countermeasures operations with autonomous underwater vehicles", *Underwater Defense Technology (UDT)*, pp. 1–6, 2005.

[100] A. R. Thompson, "Evaluating the combined uuv efforts in a large-scale mine warfare environment", DTIC Document, Tech. Rep., 2015.

[101] ARROWS, Online - http://www.arrowsproject.eu/, ARchaeological RObot systems for the World's Seas, 2012.

[102] I. Kara and T. Bektas, "Integer linear programming formulations of multiple salesman problems and its variations", *European Journal of Operational Research*, vol. 174, no. 3, pp. 1449–1458, 2006.

[103] N. Valeyrie, F. Maurelli, P Patron, J. Cartwright, B. Davis, Y. Petillot, "Nessie v turbo: a new hover and power slide capable torpedo shaped auv for survey, inspection and intervention", in *AUVSI North America 2010 Conference*, 2010.

[104] T. open source robotics foundation (OSRF), *Turtlebot2 open-source robot development kit for apps on wheels.* 2012. [Online]. Available: http://www.turtlebot.com/turtlebot2/.

[105] M. Elango, S. Nachiappan, and M. K. Tiwari, "Balancing task allocation in multi-robot systems using K-means clustering and auction based mechanisms", *Expert Systems With App.s*, vol. 38, no. 6, 2011.

[106] J. MacQueen *et al.*, "Some methods for classification and analysis of multivariate observations", in *The 5th Berkeley symposium on mathematical statistics and probability*, California, USA, vol. 1, 1967.

[107] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding", in *18th ACM-SIAM symposium on Discrete algorithms*, 2007.

[108] N. Tsiogkas, Z. Saigol, and D. Lane, "Distributed multi-auv cooperation methods for underwater archaeology", in *OCEANS 2015-Genova*, IEEE, 2015, pp. 1–5.

[109] N. Tsiogkas, V. De Carolis, and D. M. Lane, "Energy-constrained informative routing for auvs", in *OCEANS 2016-Shanghai*, IEEE, 2016, pp. 1–5.

[110] N. Tsiogkas, V. De Carolis, and D. Lane, "Towards an online heuristic method for energy-constrained underwater sensing mission planning", in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, IEEE, 2017.

[111] G. A. Croes, "A method for solving traveling-salesman problems", *Operations research*, vol. 6, no. 6, pp. 791–812, 1958.

[112] A. Singh and A. S. Baghel, "A new grouping genetic algorithm approach to the multiple traveling salesperson problem", *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, vol. 13, no. 1, pp. 95–101, 2009.

[113] H. Li and D. Landa-Silva, "An elitist grasp metaheuristic for the multi-objective quadratic assignment problem", in *Evolutionary multi-criterion optimization*, Springer, 2009, pp. 481–494.

[114] A. E. Eiben and S. K. Smit, "Parameter tuning for configuring and analyzing evolutionary algorithms", *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 19–31, 2011.

[115] N. Veček, M. Mernik, B. Filipič, and M. Črepinšek, "Parameter tuning with chess rating system (crs-tuning) for meta-heuristic algorithms", *Information Sciences*, vol. 372, pp. 446–469, 2016.

[116] M. E. Glickman, "Example of the glicko-2 system", *Boston University*, 2012.

[117] F. Li, B. Golden, and E. Wasil, "The open vehicle routing problem: Algorithms, large-scale test problems, and computational results", *Computers & Operations Research*, vol. 34, no. 10, pp. 2918–2930, 2007.

[118] N. Tsiogkas and D. M. Lane, "An evolutionary algorithm for online, resource-constrained, multivehicle sensing mission planning", *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 1199–1206, Apr. 2018. DOI: 10.1109/LRA.2018.2794578.

[119] A. Baddeley, I. Bárány, and R. Schneider, "Spatial point processes and their applications", *Stochastic Geometry: Lectures given at the CIME Summer School held in Martina Franca, Italy, September 13–18, 2004*, pp. 1–75, 2007.

[120] P. J. Diggle, J. Besag, and J. T. Gleaves, "Statistical analysis of spatial point patterns by means of distance methods", *Biometrics*, pp. 659–667, 1976.

[121] J. A. Ludwig, L. QUARTET, and J. F. Reynolds, *Statistical ecology: A primer in methods and computing*. John Wiley & Sons, 1988, vol. 1.

[122] G. J. Babu and E. D. Feigelson, *Astrostatistics*. CRC Press, 1996, vol. 3.

[123] J. Ohser and F. Mücklich, *Statistical analysis of microstructures in materials science*. Wiley, 2000.

[124] P. Elliot, J. C. Wakefield, N. G. Best, D. J. Briggs, *et al.*, *Spatial epidemiology: Methods and applications*. Oxford University Press, 2000.

[125] N. Tsiogkas and D. M. Lane, "Dcop: Dubins correlated orienteering problem optimizing sensing missions of a nonholonomic vehicle under budget constraints", *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 2926–2933, Oct. 2018. DOI: 10.1109/LRA.2018.2847719.

[126] G. Frost, D. M. Lane, N. Tsiogkas, D. Spaccini, C. Petrioli, M. Kruusmaa, V. Preston, and T. Salumäe, "Mango: Federated world model using an underwater acoustic network", in *OCEANS 2017-Aberdeen*, IEEE, 2017, pp. 1–6.

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

[128] H. Kaplan, M. Lewenstein, N. Shafrir, and M. Sviridenko, "Approximation algorithms for asymmetric tsp by decomposing directed regular multigraphs", *Journal of the ACM (JACM)*, vol. 52, no. 4, pp. 602–626, 2005.

[129] G. Dantzig, *Linear programming and extensions*. Princeton university press, 1963.

[130] R. J. Vanderbei, *Linear Programming: Foundations and Extensions*, 4th ed., ser. International Series in Operations Research and Management Science 196. Springer US, 2014.

[131] W. L. Eastman, *Linear programming with pattern constraints: A thesis*. Harvard University, 1958.

[132] J. D. Little, K. G. Murty, D. W. Sweeney, and C. Karel, "An algorithm for the traveling salesman problem", *Operations research*, vol. 11, no. 6, pp. 972–989, 1963.

[133] S. Bradley, A. Hax, and T. Magnanti, *Applied mathematical programming*. Addison Wesley, 1977.

[134] R. Gomory, "The traveling salesman problem", in *Proceedings of the IBM Scientific Computing Symposium on Combinatorial Problems*, 1966, pp. 93–121.

[135] M. Padberg and G. Rinaldi, "Optimization of a 532-city symmetric traveling salesman problem by branch and cut", *Operations Research Letters*, vol. 6, no. 1, pp. 1–7, 1987.

[136] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic", in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, IEEE, vol. 2, 1999, pp. 1470–1477.

[137] S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels, "Self-organized shortcuts in the argentine ant", *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989.

[138] J.-L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels, "The self-organizing exploratory pattern of the argentine ant", *Journal of insect behavior*, vol. 3, no. 2, pp. 159–168, 1990.

[139] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization", *IEEE computational intelligence magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[140] M. Dorigo and T. Stützle, "Ant colony optimization: Overview and recent advances", in *Handbook of metaheuristics*, Springer, 2010, pp. 227–263.

[141] F. Glover, "Tabu search part I", *ORSA Journal on computing*, vol. 1, no. 3, pp. 190–206, 1989.

[142] J. H. Holland, "Outline for a logical theory of adaptive systems", *Journal of the ACM (JACM)*, vol. 9, no. 3, pp. 297–314, 1962.

[143] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning", *Machine learning*, vol. 3, no. 2, pp. 95–99, 1988.

[144] J. H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[145] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.