UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Ronaldo Fiorilo dos Santos

Multi-robot Routing for Monitoring Areas with Different Importance Considering Failure-Prone Agents

Belo Horizonte
2023

Ronaldo Fiorilo dos Santos

Multi-robot Routing for Monitoring Areas with Different Importance Considering Failure-Prone Agents

Final Version

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Douglas Guimarães Macharet
Co-Advisor: Erickson Rangel do Nascimento

Belo Horizonte
2023

© 2023, Ronaldo Fiorilo dos Santos.
Todos os direitos reservados

Santos, Ronaldo Fiorilo dos.

S237m Multi-robot routing for monitoring areas with different importance considering failure-prone agents: [recurso eletrônico] / Ronaldo Fiorilo dos Santos. 2023.
1 recurso online (168 f. il, color.): pdf.

Orientador: Douglas Guimarães Macharet.
Coorientador: Erickson Rangel do Nascimento

Tese (Doutorado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f. 138-154.

1. Computação – Teses. 2. Sensoriamento Remoto – Teses. 3. Robótica – Teses. I. Macharet, Douglas Guimarães. II. Nascimento, Erickson Rangel do. III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação. III. Título.

CDU 519.6*82.10(043)

Ficha catalográfica elaborada pela bibliotecária Belkiz Inez Rezende Costa
CRB 6/1510 - Universidade Federal de Minas Gerais - ICEx



UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

MULTI-ROBOT ROUTING FOR MONITORING AREAS WITH DIFFERENT IMPORTANCE CONSIDERING FAILURE-PRONE AGENTS

RONALDO FIORILO DOS SANTOS

Tese defendida e aprovada pela banca examinadora constituída pelos Senhores(a):

Prof. Douglas Guimarães Macharet - Orientador
Departamento de Ciência da Computação - UFMG

Prof. Erickson Rangel do Nascimento - Coorientador
Departamento de Ciência da Computação - UFMG

Prof. Armando Alves Neto
Departamento de Engenharia Eletrônica - UFMG

Prof. Luiz Chaimowicz
Departamento de Ciência da Computação - UFMG

Profa. Mariá Cristina Vasconcelos Nascimento Rosset
Departamento de Metodologias de Computação - ITA

Prof. Tiago Pereira do Nascimento
Departamento de Sistemas de Computação - UFPB

Belo Horizonte, 11 de julho de 2023.

Documento assinado eletronicamente por **Douglas Guimaraes Macharet**, **Professor do Magistério Superior**, em 12/07/2023, às 14:39, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Armando Alves Neto**, **Professor do Magistério Superior**, em 12/07/2023, às 17:08, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Erickson Rangel do Nascimento**, **Professor do Magistério Superior**, em 13/07/2023, às 08:49, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Luiz Chaimowicz**, **Professor do Magistério Superior**, em 13/07/2023, às 11:54, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Mariá Cristina Vasconcelos Nascimento Rosset**, **Usuária Externa**, em 16/07/2023, às 22:21, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Tiago Pereira do Nascimento**, **Usuário Externo**, em 08/08/2023, às 04:28, conforme horário oficial de Brasília, com fundamento no art. 5º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufmg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2463237** e o código CRC **5FE24759**.

To my parents, Antônio Mansilha and Fátima Fiorilo.

Acknowledgments

I am thankful to my parents, Antônio and Fátima, for their consistent encouragement and support in realizing my goals. I also extend my appreciation to my brother Rafael, my sister-in-law Paula, and my cherished niece Isabela, who, despite the geographical separation, were always there.

I am deeply grateful to my beloved wife, Fernanda, for listening to my concerns and steadfastly standing by me, safeguarding my sanity, and motivating me. Without your affection and companionship, everything would have been much more difficult.

To my advisors, Douglas and Erickson, for their patience and support throughout the entire journey. While most of our interactions took place virtually, each session's conclusion left me invigorated to overcome various professional and personal hurdles. I hold immense respect for both of you as professors and, above all, as persons.

To my colleagues at UFMS, particularly professors Ivone, Juliano, Maxwell, and Rodrigo Kishi, whom I can rely on both in my professional and personal life.

To the members of VeRLab, with whom I shared experiences for slightly over a year during my UFMG internship. A special acknowledgment to Balbino, who assisted me upon my arrival in Belo Horizonte and was a research and paper writing partner. Additionally, I want to acknowledge Maurício his partnership and exceptional work in upkeeping the laboratory's infrastructure, enabling me to conduct remote experiments with minimal issues.

To the staff of the PPGCC/DCC at UFMG, especially Linda and Sonia, who have always been available to assist me with procedures, minimizing any obstacles.

To UFMS, UFMG and CAPES for enabling the Dinter program possible.

To UFMS for providing a one-year leave to carry out tasks at UFMG.

I would like to express my sincere gratitude to everyone who contributed in any way to this hard process. Thank you very much.

"*Deo gratias.*"

Resumo

O monitoramento de áreas é uma tarefa importante em diversas atividades, como em missões de busca e salvamento e no monitoramento ambiental e agrícola. Neste trabalho, estamos interessados no monitoramento de áreas com regiões com diferentes importâncias de inspeção, realizado por uma equipe de agentes heterogêneos propensos à falha. Modelamos o problema como uma instância do *Correlated Team Orienteering Problem (CTOP)* e o abordamos em duas fases: planejamento e execução da missão. Propomos um método de planejamento de rotas que se difere dos existentes na literatura principalmente por considerar que os agentes estão propensos à falha durante a execução da missão. Para tratar os casos de falha de agentes que os impedem de seguir o plano, propomos uma heurística que adapta as rotas dos agentes ativos durante o progresso da missão e busca recuperar parte da recompensa que seria perdida pela falha de um colega de equipe.

Os resultados mostram que nosso método de planejamento produz rotas pouco menos lucrativas que os métodos da literatura. Porém, nos cenários de falha dos agentes, os planos gerados pelo nosso método garantem uma menor perda de recompensa. Com relação à rotina de adaptação proposta, os resultados mostram que é vantajoso utilizá-la para recuper parte da recompensa que seria perdida pela equipe devido à falha de agentes, independente do método utilizado para gerar os planos iniciais.

Analisamos também o impacto de equipes heterogêneas no CTOP quando os agentes estão propensos à falha. Verificou-se que priorizar os agentes com recursos inferiores na fase de planejamento é mais benéfico para a equipe. Caso um desses agentes falhe, o restante dos membros da equipe possuem recursos superiores e irão adaptar suas rotas para recuperar a recompensa perdia. Porém, se esses agentes não falharem, a cobertura das regiões mais importantes de serem inspecionadas está garantida por eles.

A partir dos resultados concluímos que o método de planejamento proposto em conjunto com a rotina de adaptação de rotas são grandes colaborações para o desenvolvimento de metodologias que tratam o monitoramento de áreas no cenário de falhas dos agentes. Além disso, ao tratar de equipes heterogêneas é necessário considerar a heterogeneidade dos agentes na fase de planejamento para melhor distribuí-los no ambiente. Dessa forma, atenuamos a perda de recompensa da equipe e damos melhores condições para os agentes ativos na missão adaptarem suas rotas.

Palavras-chave: monitoramento de áreas; agentes propensos a falhas; correlated team orienteering problem

Abstract

Area monitoring is an important task in several activities, such as search and rescue missions and environmental and agricultural monitoring. In this work, we are interested in monitoring areas with different importance, carried out by a team of heterogeneous agents prone to failure. We model the problem as an instance of *CTOP* and approach it in two phases: mission planning and execution. We propose a route planning method that differs from those existing in the literature, mainly because it considers that agents are prone to failure during mission execution. To deal with cases of agent failure that prevent them from following the plan, we propose a heuristic that adapts the routes of active agents during the mission's progress and seeks to recover part of the reward that would be lost due to the failure of a teammate.

The results show that our planning method produces slightly less profitable routes than the literature methods. However, in agent failure scenarios, the plans generated by our method guarantee a smaller loss of reward. Regarding the proposed adaptation routine, the results show that it is advantageous to use it to recover part of the reward that would be lost by the team due to agent failure, regardless of the method used to generate the initial plans.

We also analyzed the impact of heterogeneous teams on CTOP when agents are prone to failure. Our findings show that prioritizing agents with lower resources in the planning phase provides greater benefits to the team. In the event of these agent failures, the remaining team members have superior resources and will adapt their routes to recover the lost rewards. However, if these agents do not fail, they still ensure coverage of the most important regions to be inspected.

Based on the results, we conclude that the proposed planning method and the route adaptation routine are great collaborations for developing methodologies that deal with monitoring areas in the scenario of agent failures. Furthermore, when dealing with heterogeneous teams, it is necessary to consider the heterogeneity of the agents in the planning phase to better distribute them in the environment. In this way, we mitigate the loss of team rewards and provide better conditions for agents active in the mission to adapt their routes.

Keywords: area monitoring; failure-prone agents; correlated team orienteering problem

List of Figures

1.1 Examples of robots performing tasks that were previously performed only by human agents. . . . 23
1.2 Illustration of autonomous agents performing area monitoring tasks in distinct activity fields. . . . 24
1.3 Aerial images of essential crops in Brazil. . . . 25
1.4 Illustration of the route adaptation for teams with fault-prone agents. . . . 27

2.1 A Correlated Orienteering Problem instance example. . . . 32
2.2 Online route replanning during mission execution. . . . 34
2.3 Concept of GTOP in an exploration scenario. . . . 40

3.1 Example of discrete environments using POIs. . . . 45
3.2 Reward and the correlation range considered in the POI utility function's calculation. . . . 46
3.3 Example of instances of the benchmark proposed by Chao *et al.* . . . 47
3.4 Examples of generated instances. . . . 48
3.5 Overview of the problem addressed in this dissertation. . . . 49
3.6 Illustration of Correlated Team Orienteering Problem with Failure-prone Agents tackled in two phases. . . . 51

4.1 Slice of an instance with a fragment from an agent's route. . . . 56
4.2 Illustration of the removals performed considering parameters $la = 15$ and $rem = 20$. . . . 58
4.3 Example of possible configurations when inserting one POI in the route 59
4.4 Illustration of insertions performed considering parameters $la = 15$ and $nr = 1.5$ 60
4.5 Illustration of the route segment before and after the execution of Algorithm 1 61
4.6 Baseline deployment scheme of our adaptation routine in a real area monitoring mission . . . 65
4.7 An illustrative example of an instance with 225 POIs with one region more important to inspect at the center . . . 68
4.8 Comparison for a team of three agents in an instance with 225 POIs when one agent fails in a certain percentage of the plan . . . 70
4.9 Comparison for a team with three agents in an instance with 225 POIs when two agents fail . . . 72

4.10 Comparison for a team with ten agents in an instance with 900 POIs when up to nine agents can fail . . . 73
4.11 Example of unbalanced routes: instance with one region more important to inspect at the center. . . . 74
4.12 Example of unbalanced routes: instance with two regions more important to inspect at the upper corners . . . 74
4.13 Region Specificity Levels for an environment with 30 POIs. . . . 76
4.14 Process of dividing the environment into regions. . . . 77
4.15 Estimating reward and costs for a more generic level of representation. . . . 78
4.16 Optimality gap of the solution computed with Gurobi . . . 87
4.17 Comparison of initial plans obtained according to the number of POIs in the instance. . . . 88
4.18 Comparison of final reward collected when one agent fails according to the number of POIs in the instance. . . . 89
4.19 Direct comparison between instances using MIQP against our planning method 90
4.20 Instance with optimal solution calculated by the MIQP method . . . 92
4.21 Direct comparison between instances using MIQP against our planning method considering only teams with two agents . . . 93
4.22 Direct comparison between instances using MIQP against our planning method considering only teams with three agents . . . 94
4.23 Direct comparison between instances using MIQP against our planning method considering only teams with four agents . . . 94

5.1 Example of commercial drones. Images are not on the same scale. . . . 98
5.2 Example of commercial ground robots. Images are not on the same scale. . . . 100
5.3 Comparison of computed rewards considering all teams with four members . . 108
5.4 Percentages of rewards lost with the failure and recovered with the adapting routine . . . 109
5.5 Comparison of routes generated for a homogeneous team and for a heterogeneous team . . . 110
5.6 Analysis of the routes generated for the picked heterogeneous teams . . . 111
5.7 Comparison of the computed average rewards considering the four spotlighted teams, according to the number of POIs in the instance. . . . 112
5.8 Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance . . . 113
5.9 Comparison of computed rewards considering all teams with four members . . 116
5.10 Percentages of rewards lost with the failure and recovered with the adapting routine . . . 117
5.11 Comparison of computed rewards considering all teams with four members . . 121

5.12 Analysis of the routes generated for the TEAM-4 123
5.13 Comparison of computed rewards considering all teams with four members . . 125
5.15 Comparison of computed rewards considering all teams with four members . . 129
5.16 Percentages of rewards lost with the failure and recovered with the adapting routine 130

A.1 Percentages of rewards lost with the failure and recovered with the adapting routine 147
A.2 Comparison of routes generated for a homogeneous team and for a heterogeneous team 149
A.3 Analysis of the routes generated for the picked heterogeneous teams 150
A.4 Comparison of the computed average rewards considering the four spotlighted teams, according to the number of POIs in the instance. 151
A.5 Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance 152

B.1 Percentages of rewards lost with the failure and recovered with the adapting routine 154
B.2 Comparison of routes generated for a homogeneous team and for a heterogeneous team 156
B.3 Analysis of the routes generated for the picked heterogeneous teams 156
B.4 Comparison of the computed average rewards considering the four spotlighted teams, according to the number of POIs in the instance. 157
B.5 Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance 158
B.6 Percentages of rewards lost with the failure and recovered with the adapting routine 159
B.7 Percentages of rewards lost with the failure and recovered with the adapting routine 160
B.8 Comparison of routes generated for a homogeneous team and for a heterogeneous team 161
B.9 Analysis of the routes generated for the picked heterogeneous teams 162
B.10 Comparison of the computed average rewards considering the four spotlighted teams, according to the number of POIs in the instance. 163
B.11 Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance 164
B.12 Percentages of rewards lost with the failure and recovered with the adapting routine 165
B.13 Analysis of the routes generated for the picked heterogeneous teams 167

B.14 Comparison of the computed average rewards considering the four spotlighted teams, according to the number of POIs in the instance. 167
B.15 Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance 168
B.16 Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance 169

List of Tables

2.1 Main works found in the literature and the characteristics addressed by them, contextualizing the problem proposed in this dissertation proposal. 43

4.1 Summary of the main functions complexity of Algorithm 1 depending on the elements involved and, in the worst-case scenario. 63
4.2 Comparison between the initial plan obtained by the GA and the one optimized by our method 69
4.3 Comparison of different values of la and nr for a team with three agents in an instance with 900 POIs 71

5.1 Some commercial multi-rotor drone models. 99
5.2 Some fixed-wing drone models. 99
5.3 Some ground robot models 101
5.4 Classification of robots used in the analysis of heterogeneous teams. 103
5.5 Classification of sensors used in the analysis of heterogeneous teams. 103
5.6 Considered robots in the experiments with heterogeneous budgets 106
5.7 Considered robots in the experiments with heterogeneous perception 115
5.8 Considered robots in the experiments with heterogeneous speeds 120
5.9 Considered robots in the experiments with heterogeneous sensing times 124
5.10 Considered robots in the experiments with heterogeneous costs 127

A.1 Considered robots in the experiments with heterogeneous perception 148

B.1 Considered robots in the experiments with heterogeneous speeds 155
B.2 Considered robots in the experiments with heterogeneous sensing times 160
B.3 Considered robots in the experiments with heterogeneous costs 165

List of Acronyms

ALNS	Adaptive Large Neighborhood Search
AOPST	Adaptive Orienteering Problem with Stochastic Travel Times
CluOP	Clustered Orienteering Problem
CluTOP	Clustered Team Orienteering Problem
COP	Correlated Orienteering Problem
COPTW	Cooperative Orienteering Problem with Time Windows
CSP	Constraint Satisfaction Problem
CTOP	Correlated Team Orienteering Problem
CTOP-FPA	Correlated Team Orienteering Problem with Failure-prone Agents
GA	Genetic Algorithm
GTOP	Generalized Team Orienteering Problem
IPP	Informative Path Planning
KP	Knapsack Problem
LNS	Large Neighborhood Search
MILP	Mixed Integer Linear Programming
MIP	Mixed Integer Program
MIQP	Mixed Integer Quadratic Programming
MSA	Multi-start Simulated Annealing
OP	Orienteering Problem
OPSTS	Orienteering Problem with Stochastic Travel and Service Times
OPSW	Orienteering Problem with Stochastic Weights
OSOP	Operational-level Stochastic Orienteering Problem

OVRP	Open Vehicle Routing Problem
POI	Point Of Interest
POIs	Points Of Interest
POP	Probabilistic Orienteering Problem
RGB	Red, Green and Blue
RMOP	Robust Multiple-Path Orienteering Problem
SR	Standard Robot
STSP	Selective Traveling Salesman Problem
SA	Simulated Annealing
SAA	Sample Average Approximation
SOM	Self Organizing Map
STOP	Steiner Team Orienteering Problem
S-TSPN	Sampling Traveling Salesperson Problem with Neighborhoods
TOP	Team Orienteering Problem
TOPTW-MV	Team Orienteering Problem with Time Windows and Mandatory Visits
TSOP	Team Surviving Orienteers Problem
TSP	Traveling Salesman Problem
UAV	Unmanned Aerial Vehicle
UAVs	Unmanned Aerial Vehicles
UGV	Unmanned Ground Vehicle
RSL	Region Specificity Levels

List of Symbols

β_k	Available budget for the k-th agent
π	Route with starting point in v_{orig} and arrival point in v_{dest}
$\pi^{[b:e]}$	Segment of the π route from point v_b^π to point v_e^π
$\boldsymbol{\Phi}$	Set of routes found for the agents
Ψ	Space of all possible routes
$\boldsymbol{\mathcal{A}}$	Set of agents that integrate the team
a_k	k-th agent of $\boldsymbol{\mathcal{A}}$
$\mathcal{C}(\cdot)$	Cost function of collecting the reward at a point v_i
c_{rwd}	Correlated reward between points
cr	Maximum distance of the points considered when calculating the extra reward
$\mathcal{D}(\cdot)$	Cost function of moving from one point to another
$dist(\cdot)$	Distance function between two points
la	Size of the analyzed segment in the adaptation routine
$\boldsymbol{\mathcal{N}}_{v_i}^{cr}$	Neighborhood of point v_i delimited by the correlation range cr
$\boldsymbol{\mathcal{N}}_{v_i}^{nr}$	Neighborhood of point v_i delimited by the neighborhood range nr
nr	Maximum distance of the points considered in the adaptation routine when choosing a point to be inserted in the route
o_i	Spatial location of point v_i in $\mathbb{R}^2$
pos	Current position of the agent
rem	Percentage of points removed from the analyzed segment in the adaptation routine
r_i	Reward received when visiting the point v_i
$\boldsymbol{\mathcal{V}}$	Set of points to be sensed
v_{dest}	Destination point of all routes
v_i	i-th point of $\boldsymbol{\mathcal{V}}$
v_i^π	i-th point of the π route
v_{max}	The node that maximizes the reward in the adaptation routine
v_{orig}	Origin point of all routes
w_{ij}	Indicates the amount of information v_i can get from its correlation with v_j

x_i	Variable that indicates if node v_i is visited in one of the solution routes.

Contents

1 Introduction **21**

1.1 Context . . . 22
1.2 Motivation . . . 23
1.3 Problem . . . 25
1.4 Contributions . . . 28
1.5 Text Organization . . . 29

2 Related Work **30**

2.1 Correlated Orienteering Problem (COP) . . . 32
2.2 Stochastic elements . . . 34
2.3 Regions with different importances . . . 36
2.4 Teamwork and heterogeneous agents . . . 38
2.5 Two-stage approaches . . . 39
2.6 Problem contextualization . . . 42

3 Methodology **44**

3.1 Environment representation . . . 45
3.2 Problem Formulation . . . 48

4 Homogeneous teams **53**

4.1 Planning Phase . . . 53
4.2 Execution Phase . . . 54
4.3 Planning for failure-prone teams . . . 75
4.4 Concluding remarks . . . 95

5 Heterogeneous teams **97**

5.1 Agents heterogeneity . . . 97
5.2 Mission planning and execution . . . 104
5.3 Concluding remarks . . . 132

6 Conclusions and Future Works **134**

6.1 Conclusions . . . 134
6.2 Future works . . . 138

Bibliography **139**

A Heterogeneous perception **147**

B Heterogeneous costs **154**

B.1 Different speeds 154

B.2 Different sensing times 159

B.3 Different speeds and sensing times 165

Chapter 1

Introduction

Nowadays, the diversity of robots in the world is vast. There are countless options with different physical characteristics and diverse ways to interact with the environment [15]. An exciting category of robots is the autonomous agents, which encompasses robots capable of performing real-world tasks alone, without direct human control. Due to the various sensors capable of collecting information from the real world and the advancement of artificial intelligence, more and more autonomous agents are being used to perform society's daily tasks. Among these tasks, area monitoring is receiving much attention due to the success obtained when several types of robots are used to perform it [12, 32, 33].

Generally, area monitoring involves one or more robots collecting data from predetermined target locations. Using multiple robots can bring several advantages, like increased robustness and, in most cases, reduced time required to accomplish a given task. In contrast, it also raises many challenges, such as planning, coordination, and control [49]. Furthermore, most works in literature usually consider static instances, i.e., the robots are dispatched to follow an unchangeable and previously planned set of routes. However, such an assumption is too restrictive and may not account for the dynamic nature of many real-world scenarios, for example, unexpected failures that might occur during the monitoring [46].

This dissertation presents the modeling of a route planning system for monitoring missions in regions with different monitoring importance. We consider that this task will be performed by a team of robots, where a subset of them may fail during the mission execution. If a robot does not comply with the plan, the system adapts the routes of the active robots in the mission to minimize the damage to the team's final goal caused by the failures.

1.1 Context

Many routine societal activities involve monitoring or inspection tasks to verify if something is following the desired, such as inspecting construction sites in civil engineering [20] or monitoring animals [73] and crops in agriculture [17]. Among these routines include areas monitoring, such as the essence of surveillance and tracking tasks [77], search and rescue missions [3], and environmental and agricultural monitoring [60, 54], among others [10, 58]. In such activities, one or more agents navigate through an environment to get information about it or look for a specific target or feature. In addition, in environments that need to be monitored, we usually have regions that are more important to be inspected than others and deserve greater attention from agents.

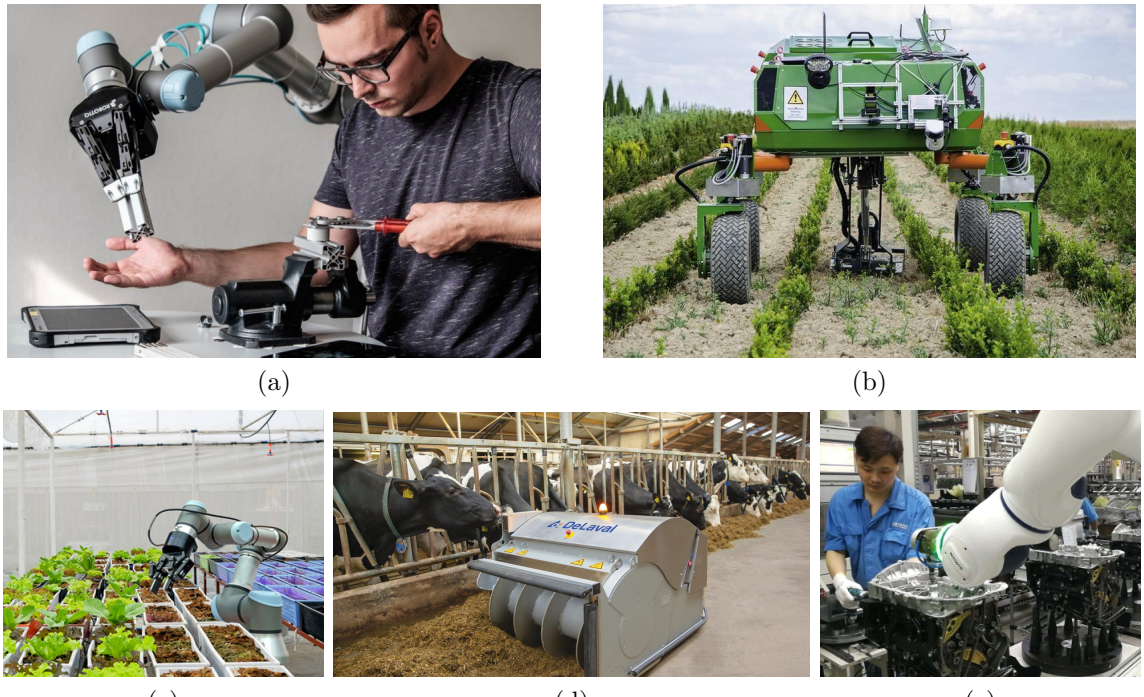
For a long time, such tasks were performed only by human agents, who manually oversaw interest areas. The problem with this practice is conditioning the monitoring quality to the human cognitive load. Besides, when the surveyed area has many points of interest, a human agent can neglect some regions due to momentary inattention or lack of action in those regions [59]. With the advent of affordable robotics and its application to the most diverse real-world tasks, autonomous robotic agents have also been successfully used in area monitoring tasks. One of these agents' advantages is the concentration expended in the task execution without interference from factors such as tiredness, inattention, or others that may affect the task's final result [38]. Furthermore, a wide variety of robots and sensors allow safe access to areas at risk for humans and capture information from several points of view.

The idea of having autonomous agents performing these tasks become more interesting when we consider using a set of robots working collaboratively, which implies increased performance and reliability in mission accomplishment. Planning an area monitoring mission for a team of robots is challenging. We must respect the restrictions and make the most of each agent's characteristics to determine which environment region is best for a robot to cover. Due to the different ways of capturing information from the world, we must merge this diverse data to meet the mission objective fully [1, 2].

In a real-world area monitoring mission, some factors can interfere with the planned actions for the robots, such as sensor failures, motion problems, passage blockages in the environment, and others that prevent them from following the plan. Considering that unforeseen events can occur during mission execution includes a degree of uncertainty arising from the robot's performance that directly interferes with the entire team's planning. In this way, it is necessary to develop techniques that enable autonomous agents to perceive changes in the execution of the team's plan and adapt its plan to maximize the team's success in the mission.

1.2 Motivation

The interest in using robots in activities previously performed only by human agents has grown considerably. In many applications, autonomous agents have aided or even replaced humans, as seen in Figure 1.1. The reasons for this increase are the reduction of involved costs in various activities, more agility in performing repetitive tasks, the absence of human factors such as tiredness and inattention, and especially a lower human exposure to situations that pose some risk [30].



Images sources: (a) Analytics India Magazine (b) Fraunhofer Institute for Manufacturing Engineering and Automation (c) Association for Advancing Automation (d) Dairy Herd Management (e) International Federation of Robotics

Figure 1.1: Examples of robots performing tasks that were previously performed only by human agents. (a) assistance in assembling parts; (b) weed control in agriculture; (c) seedling management in the greenhouse before planting in soil; (d) feed pusher for cows; and (e) assistance in an industrial assembly line.

Autonomous agents have been increasingly used in the area monitoring task due to the possibility of accessing hard-to-reach regions for humans and areas with risk of collapse or other hazards [63]. Also, some robots, like aerial vehicles, can cover vast land ranges with relatively little time, which allows them to obtain information more efficiently in various situations. Figure 1.2 illustrates Unmanned Aerial Vehicles (UAVs) applications in monitoring tasks in agriculture, mining, and disaster areas. The illustration shows only one UAV acting on each scenario in these examples. However, using a team of collaborative autonomous agents is a theme that has been growing with great importance

in mobile robotics and brings several advantages compared to systems composed of a single agent [24]. For example, we can use air and ground agents working collaboratively to achieve a common goal, delegating different functions to each type of agent. Thus it is possible to arrive at the task's fulfillment more efficiently. Besides, the system becomes more robust to individual failures with more agents available.



Image source: Aero Engenharia - Sensoriamento Remoto

Figure 1.2: Illustration of autonomous agents performing area monitoring tasks in distinct activity fields. From left to right: agriculture, mining, and disaster area.

In this dissertation, we are interested in monitoring areas with regions that are more important to inspect than others, using a set of collaborative agents. Completing an area monitoring task is usually divided into two phases: the planning and execution of the mission, in that order. Most literature addresses this task using a set of deterministic agents for both steps, where the agents strictly follow what was planned in the first phase during the execution of the mission. Here, we are interested in considering agents prone to failure during mission execution. In this phase, a failure can prevent the agent from fulfilling what was assigned in the planning stage and compromise the team's outcome. This premise is valid in real-world situations, as sensor failures can occur without giving any previous signals, and the agent can have locomotion problems or any other crash.

Including collaborative agents and assuming that team members may fail when solving area monitoring problems is essential to obtain a scenario closer to the real world. Considering the agents' diverse capabilities, we can extract the maximum from each one, maximizing the information collected in the mission. When one or more agents are prone to failure and risk not accomplishing what was planned, we have to provide mechanisms for the other autonomous agents to realize this disagreement and take some actions to ensure the mission's success. This step is essential in environments with regions that are more important to inspect than others. The efforts of the autonomous agents still active in the mission can determine the team's success in achieving the goal and ensure that key

areas of the environment are examined.

We can see a real-world application of the problem tackled in this dissertation in monitoring agricultural areas. Important crops in Brazil, such as soybeans, sugar cane, and planted eucalyptus forests, occupy large tracts of land, as shown in Figure 1.3. For a profitable harvest, constant monitoring in these areas is necessary to search for plant anomalies that may harm production. Combining ground and aerial agents with several sensors, such as multi-spectral or high-resolution cameras, lasers, and soil analyzers, can collect useful information for understanding the environment and detecting changes in sites [47]. In scenarios like these, some regions are more important to monitor that others, for example, those in a specific growth stage or under some particular treatment.



(a) (b) (c)

Images sources: (a) CNN Brasil (b) GestAgro 360° (c) A Lavoura Magazine

Figure 1.3: Aerial images of essential crops in Brazil. (a) soybean; (b) sugar cane; and (c) eucalyptus forest.

1.3 Problem

Area monitoring tasks usually require the investigation of large areas, as is the case, for example, of agricultural monitoring. In these environments, agents must monitor several hectares of the planted crop, and success in carrying out the task is crucial for a good harvest in the future. Ideally, the best plan would be to visit and analyze the entire planted area. Executing this plan would be cost-prohibitive due to a large amount of Points Of Interest (POIs) to sense, rendering it impractical. However, these environments have a characteristic enabling obtaining of correlated information from one Point Of Interest (POI) when visiting another. This is because, on a plantation, we rarely have a single plant that is sick or attacked by pests among the other plants nearby. Also, in such contexts, some regions are more important to inspect than others. For example, areas that have been receiving some treatment or are close to others that have recently suffered a pest attack. Once stated these observations, an effective plan does not need to visit the entire area but rather POIs that allow obtaining correlated information from its vicinity,

prioritizing those POIs within the most important regions. A plan with these features provides an overview of the entire environment, mainly of the greatest interest regions.

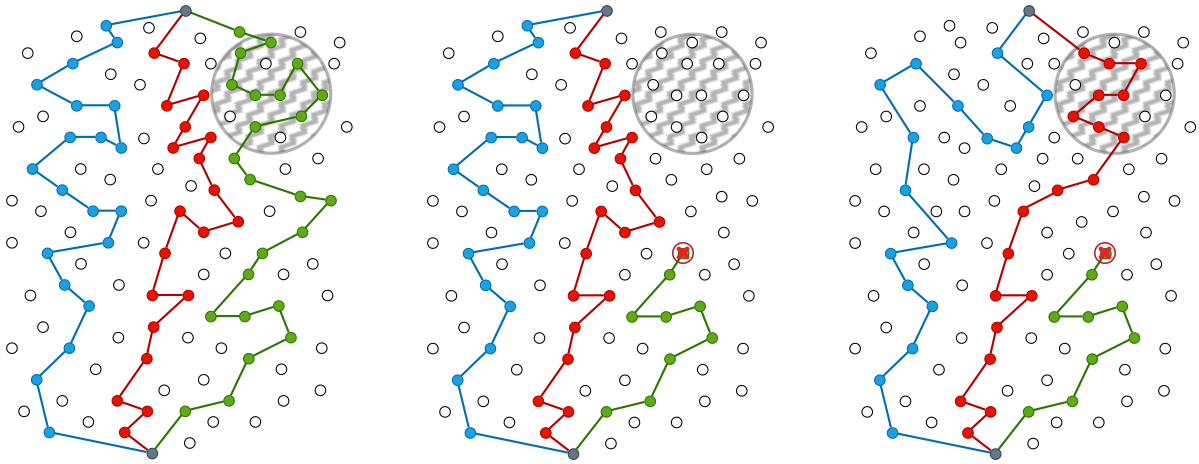
Using a team of agents with different capabilities allows for sensing the environment from different perspectives. Some agents are better suited than others to visit certain regions of the environment according to the sensors they are equipped with. For example, we can use aerial agents to get an overview of the situation, while ground agents visit critical regions of the environment to get more detailed information about it. Also, in areas that are difficult to access, it may be that only an aerial agent can get some information. Ideally, route planning should consider the particularities of each agent to decide in which regions these agents should act for better collaborative monitoring of the area of interest.

Solutions to routing problems are usually designed based on solutions developed for the well-known Traveling Salesman Problem (TSP) [41]. The TSP consists of determining the shortest route through a series of POIs scattered in the environment, visiting each point only once, and returning to the POI of origin. By itself, the TSP is already a difficult problem to be solved. We have a more challenging mission in the problem we are addressing because we consider multiple agents. Furthermore, we are interested in areas with regions more important than others to be sensed. In a real-world application, we must assume that each agent does not have unlimited resources to accomplish the mission and must respect a pre-defined budget. This budget constraint prevents the agents from visiting all the POIs, making a need to choose which ones to visit.

The vast majority of routing problem approaches are concerned with only getting path planning for each agent, not considering the progress of the mission and possible agents' failures. When dealing with real-world scenarios, we must consider the possibility of failure, which introduces a level of uncertainty about the exact execution of the planning by the agents. For example, when an agent fails and does not follow the planned route, it compromises the team's final result because it is not visiting the important regions assigned to it. When this occurs, we need a mechanism that can adapt other agents' routes to decrease the impact of these losses on the initial plan. It is precisely in this process of considering that one or more agents may not follow the plan and provide a method of adapting the routes of the agents that this dissertation aims to contribute. Following this, we define the problem addressed in this dissertation.

Multi-robot Routing for Monitoring Areas With Different Importance Considering Failure-Prone Agents: *Given a set of points of interest where each point offers a reward to be visited and a correlated reward from the nearby unvisited points — consider the existence of subsets of nearby points that provide greater rewards than the others. Given a set of agents with different capabilities and a predetermined budget for each. Considering an unknown subset of agents that can fail in the mission execution. Plan one route for each agent, respecting the agent's budget, in order to maximize the team's final rewards collection even if agents do not follow the plan to the letter.*

Figure 1.4 illustrates the problem with an instance composed of 119 POIs spread across an environment, where one region is more important to sense than others, which is indicated by the hatched circle at the top right of the POIs set. Figure 1.4a shows the initial planning with all routes starting at the lowest gray point and finishing at the top gray one. During the plan execution, we consider that an agent can fail at any time. The paths taken by the agents at the end of the mission execution are shown in Figures 1.4b and 1.4c. In these figures, the red check interrupting the green route indicates that the agent failed at that point and abandoned the mission. If the remaining active agents in the mission ignore the teammate's failure, they will follow the routes assigned to them in the initial planning, as shown in Figure 1.4b. This scenario indicates that the failure of the green agent drastically impacts the team's mission success, as any team member does not even visit the region that was the most important to be sensed.



(a) (b) (c)

Figure 1.4: Illustration of the route adaptation for teams with fault-prone agents. (a) initial route planning for a team with three agents; (b) the green agent fails at the red check and abandons the mission; and (c) the green agent fails and abandon the mission, but the other agents adapt its courses to try to cover part of the area abandoned by the green agent.

Attempting to reduce the impact caused by the failure of the green agent, Figure 1.4c shows the paths executed when the other members have their routes adapted in response to this failure. We can note that the other two agents adapted their courses to try to cover part of the region that the failed agent would no longer visit. Adjustments must keep the route's cost below the agents' budget limit. Therefore, it is necessary to develop techniques that, when identifying that some agent has failed, can adapt the other agents' routes to reduce the impact caused by this failure, taking into account the budget limit of each agent.

In this dissertation, we addressed the problem of monitoring areas with different importance as a Correlated Team Orienteering Problem (CTOP), which is discussed in

detail in Chapter 2. We chose the CTOP because it is a recently presented problem that fine models the issue, and we believe we can contribute with extensions to its formulation. Besides considering that the routes must be generated respecting a budget limit for the agents, CTOP has the characteristic of taking into account information from neighboring points of the visited POIs, which fits the areas monitoring tasks where the POIs are correlated. In this thesis, we extended CTOP to handle failure-prone agents not complying with the plan during mission execution. For this purpose, we developed a planning method to generate routes seeking to minimize agent failures' impact. We also propose a route adaptation routine for active agents in the mission, aiming to recover part of the information that would lose with any agent's failure.

1.4 Contributions

The treatment of agent failures involving CTOP with heterogeneous teams is still underexplored in the literature. The main contributions obtained from this work are presented, grouped according to the different challenges related to the problem are:

- **Route adaptation**: Development of an online strategy to efficiently adapt routes in a planned area monitoring mission as an instance of CTOP. The method proved effective in recovering rewards lost due to agents' failures during mission execution, regardless of the method used to plan the routes. Quantitatively, the value of the final rewards is slightly lower than using a route replanning method. However, the time spent and the fact that the adaptations are made without the need to keep the agents stationary are significant contributions.

- **Route planning considering failure-prone agents**: A route planning methodology was presented that divides the environment into regions and ensures that more than one agent passes through regions that are more important to be inspected. The main differential of the method is that it is concerned with maximizing the reward collected by the team and minimizing the loss in case of agent failure, and facilitating the proposed adaptation routine. Due to this, the missions are slightly less profitable than those generated by methods in the literature, but when failures occur, the loss of reward is considerably lower. Additionally, this method provides a notable increase in speed compared to existing methods in the literature.

- **Treatment of heterogeneity in teams**: The route planning and adaptation methods were extended to address agent heterogeneity. The impact of three types

of heterogeneity was analyzed: uptime autonomy, movement and sensing costs, and perception capability. Although there are methods in the literature that address the heterogeneity of agents with CTOP, to the best of our knowledge, none specifically consider it in the context of failures. The analyses indicated that prioritizing agents with lower resources to visit more important regions to be inspected is more beneficial for the team. In case these agents fail, the team still has agents with better resources to adapt their routes and recover the reward. And if they do not fail, the coverage of important points will be guaranteed.

The route adaptation methodology and the discussion of the obtained results were published in the proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA 2021).

- Ronaldo F. dos Santos, Erickson R. Nascimento, Douglas G. Macharet. Anytime Fault-tolerant Adaptive Routing for Multi-Robot Teams. *International Conference on Robotics and Automation (ICRA)*, 2021.

The proposed planning for failure-prone agents and the analysis of the impact of heterogeneous teams were reported in a paper submitted and under review to a journal.

1.5 Text Organization

This document is organized into five chapters in addition to this Introduction. Chapter 2 presents a review of the main works found in the literature related to the problem we are addressing. In Chapter 3, we describe how we represent the environments and formulate the problem in a general way. Chapters 4 and 5 present the proposed method for planning routes and making adjustments to agents' routes in case of failures during mission execution. Chapter 4 considers teams with homogeneous agents, while in Chapter 5, we analyze the impact of considering heterogeneous teams in monitoring missions with agents prone to failure. Finally, in Chapter 6, we summarize the results obtained, highlight the advances made in solving the problem and point out directions for future work that can be started from what was developed in this dissertation.

Chapter 2

Related Work

In many real-world situations, exploring an environment to obtain information about it is necessary, such as area surveillance, search and rescue missions, and agricultural monitoring. For better accomplishment in these tasks, using a team with collaborative agents for the mission is preferable rather than a single agent. This dissertation considers that among the teammates, there are some failure-prone agents that, at some point, may not fulfill the actions provided for the initial plan, compromising the team's final result.

Area monitoring tasks require agents to traverse an environment to get interesting information. Ideally, an agent should cover and analyze the full area extension to inspect adequately. However, this practice is only feasible when the monitored environment has few points of interest or when agents have unlimited resources, which is difficult to happen in real-world situations. In this way, the area monitoring problem is frequently interpreted as the challenge of capturing as much information as possible from the environment as opposed to maximal area coverage. This task requires computing a sequence of sensing locations to be visited that maximize the gain of information an agent acquires according to an appropriate gain function.

Another well-known problem with this purpose and widely used to model the areas monitoring task is called Informative Path Planning (IPP) [50, 44]. Although the main approaches to the IPP solution represent the environment in a continuous way, it is also possible to approach the problem by discretizing the environment in a finite amount of points (nodes) and representing it as a graph. In this dissertation, we are interested in the approaches that discretize the environment to plan the team routes.

Determining routes that are either length or time optimized for autonomous agents is an important task. It has been the focus of several research fields, such as robotics, optimization, and heuristic design [43, 46]. In the context of discretizing the environment and representing it as a graph, the Traveling Salesman Problem (TSP) [41] is one of the most classic problems, and since it is NP-Hard, several heuristics to tackle it have been proposed [6]. Despite remarkable advances in determining routes using TSP, this classical formulation might be insufficient for different scenarios, thus demanding its generalization.

One of these generalizations was presented by Laporte and Martello [42], being called Selective Traveling Salesman Problem (STSP), where the objective is to determine

a path restricted by a budget that visits a group of points, maximizing the sum of collected rewards. Chao *et al.* proposed a generalization similar to STSP, called Orienteering Problem (OP) [18]. The OP can be seen as the combination of the Knapsack Problem (KP) [37] when determining the subset of points to be attended, and the TSP, choosing the order to visit these points. The inspiration for OP came from orienteering sports, where some control points are scattered on a map, and a player must physically visit these places to collect the provided reward. The goal for a player is to plan the best path to sum the maximum of rewards accumulated in a predetermined time.

Multiple variants and generalizations were proposed from OP to incorporate other aspects of real-world problems, as Feillet *et al.* [23] and Vansteenwegen *et al.* [70] show it. A direct extension of the OP classic formulation is to consider using multiple agents working as a team. This variation is known as Team Orienteering Problem (TOP) [19], and its solution becomes a set of routes, one for each team agent, with the same start and endpoints. At the TOP, the goal is that the paths planned for each agent accumulate as much reward as possible when the mission ends. Moreover, this planning must consider each agent's specifics to ensure the plan's feasibility.

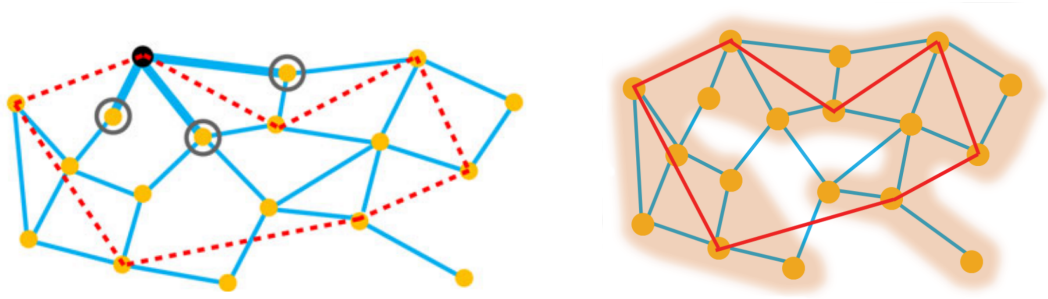
Our work is based on a variant of OP known as Correlated Orienteering Problem (COP) [75, 76] . We present a more detailed discussion of this variant in Section 2.1. Since COP and its version for multiple agents Correlated Team Orienteering Problem (CTOP) are relatively new problems, not many studies incorporate the hypothesis that some agents may fail in their formulations. This is a very important element when planning real-world exploration missions because it is a way to represent the inherent uncertainty of acting in real-world environments in modeling the problem. Considering that the key and crucial difference between the COP and the OP is that the COP assumes that the neighboring nodes have correlation and the OP does not, we will discuss some works that address the OP, bearing in mind that an adaptation for the COP is made straightforwardly. Among the numerous OP variants, we will discuss some works that we believe have a closer relationship with the problem of monitoring areas with different importance to inspect and with what we intend to propose to approach it.

To better organize the text, we divided this Chapter into sections according to what we incorporated into our approach. Section 2.1 presents the COP and the CTOP jointly with works that revolve around them. We addressed the uncertainty that the agent would complete the plan since possible failures. The uncertainty is usually represented using some stochastic element in routing problems. OP variants that consider stochasticity are discussed in Section 2.2, focusing on possible agent failures. Section 2.3 raises works considering instances with must-visit nodes to approximate the idea of areas with different importance to inspect. Section 2.4 discusses works that deal with teammates with different capabilities, while works with a two-phase approach are discussed in Section 2.5. Finally, in Section 2.6, we contextualize our current work within the literature.

2.1 Correlated Orienteering Problem (COP)

The Orienteering Problem (OP) and its variants have been used in several applications to model sampling or information-gathering tasks. It was observed that the information collected at the nodes is frequently correlated between nodes close to each other. Considering this observation, Yu *et al.* [75, 76] proposed an extension for OP considering that when visiting a given node, it is possible to get partial information about its neighbors. This extension was called Correlated Orienteering Problem (COP), or in the case of multi-agent assembles Correlated Team Orienteering Problem (CTOP). In addition to formulating the problem as a nonlinear extension of OP, the authors provided Mixed Integer Quadratic Programming (MIQP) models to solve the problem for single and multiple agents.

Figure 2.1 illustrates a COP instance example. Both images represent the same scenario. The nodes of the graphs represent the points of interest in the environment where sensing is performed. The agent is allowed to move from one node to any other node; the edges present in the graphs indicate only the correlation of information between the connected nodes (Figure 2.1a). In this example, the authors were interested in finding routes with identical start and end nodes, i.e., a cycle.



(a) (b)

Figure 2.1: A COP instance example. (a) When a node is sensed by an agent (the black node), it is possible to obtain partial information from its neighbors due to the spatial correlation between the nodes connected by edges (the three nearby, circled nodes) [76]; (b) Following the indicated path in red, it is possible to obtain at least partial information from all nodes of the graph [75].

Using correlations between nodes allows inferring at least partial information for nodes not directly visited by the agent, enabling a broader knowledge of the sensed environment, even if it is incomplete. Figure 2.1b reinforces this statement with all graph nodes inside the shaded region, indicating where it was possible to obtain some information by executing the path highlighted in red. This feature makes COP an interesting approach for the areas monitoring task because even though it is impossible to cover the

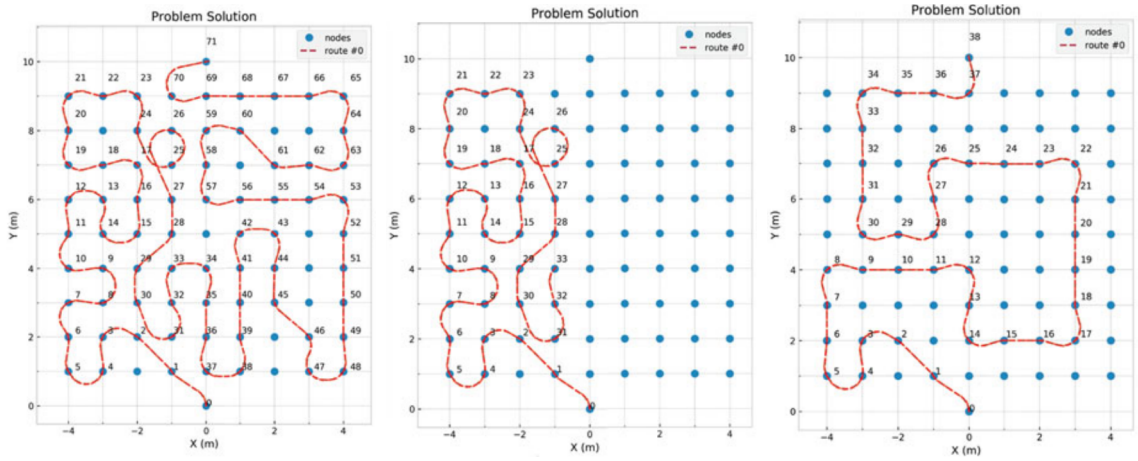
entire environment, it can infer information from regions correlated to those visited by the agent.

In this sense, Tsiogkas *et al.* addressed mission planning for scientific exploration in underwater environments, modeling the problem as a COP instance [64]. The goal was to gather as much information as possible without exceeding the maximum amount of autonomous marine vehicle energy. For this scenario, the authors compared the solution obtained for the COP with solutions obtained for the OP and the Open Vehicle Routing Problem (OVRP). They reported that the quality of the paths generated using the COP was better because the COP sought to spread the route over the whole environment. Regarding the time complexity, the MIQP method to solve the COP was the most expensive and impractical to use in large instances.

Given this disadvantage of the complexity of solving the problem exactly, Tsiogkas *et al.* proposed, as they said, the first heuristic to solve the COP [65]. The authors showed that the genetic algorithm-based heuristic could find a solution in a time well below that using the MIQP method while the quality of the solutions remained high. The results obtained by the heuristic were compared with two different executions of the MIQP solver, with 1% and 5% of optimality gap. As expected, the execution of the MIQP, whose output is 1% different from the optimum, generated a route with greater reward. The heuristic performance was better than the 5% MIQP solution, showing a very close route quality. Concerning the time it takes to find a solution, the heuristic performs orders of magnitude lower than the MIQP solutions. This gain in execution time becomes more clearly visible in larger instances of the problem.

Arora and Scherer proposed an algorithm to solve the COP by breaking the problem into finding the set of nodes that should be visited and then defining the optimal order in which these nodes should be visited [7]. They formulated the set selecting problem as a Constraint Satisfaction Problem (CSP) and found the optimal order of visiting these nodes as a TSP. The space of routes feasible for the solution is explored using a greedy local search heuristic to improve execution time. Also, in a step where nodes are sampled to enter or leave routes, this sampling is weighted to choose the most valuable nodes often. The results presented show solutions with at least 95% of the value of the optimal solution and a much shorter execution time of the state-of-the-art method. However, according to the authors, the method is still unacceptably slow for large instances.

Tsiogkas and Lane tackled the online task scheduling problem for sensing missions performed by multiple vehicles [66]. They proposed a heuristic to solve CTOP based on a Genetic Algorithm (GA). The reported results show that the utility captured near to the optimal MIQP solution, proposed by Yu *et al.* [76], and about 300 times more efficient in time execution. However, the studied instances had less than a hundred nodes, and the authors not reported experiments for online usages, such as route readjustment. Later, Tsiogkas and Lane used COP with Dubins vehicles and performed more experiments



(a) (b) (c)

Figure 2.2: Online route replanning during mission execution. (a) Full route that the agent is supposed to follow; (b) Where the mission would end if no rescheduling were done; (c) The redesigned route considering the new budget [67].

considering that the budget consumption changes along with the mission execution [67]. They reported results for a single agent in a 9×9 grid, where after completing 10% of the mission, the agent notices that it has consumed 20% of its budget and will not be able to complete the initial plan. Figure 2.2 illustrates this situation, where we can see that the re-planned solution allows better coverage of the surveyed area.

The literature covering the CTOP and its applications is still recent and, therefore, limited. In the next Sections, we discuss works that deal with OP variants and have characteristics we try to implement at the CTOP.

2.2 Stochastic elements

When dealing with real-world problems, many unforeseen events may occur and ruin what was planned. Therefore, considering the stochasticity of elements when planning a monitoring mission allows us to generate plans that are more reliable with reality and robust to deal with any surprises during the execution of the mission. Several studies describe variants of OP with uncertainties associated with some element of the problem [26, 69]. The main components chosen to be treated as stochastic are the reward offered by a node [31, 72], the travel time between one node and another [22, 71], and the service time required to receive the reward [27, 28]. Some problems combine more than one stochastic element in their formulation [11, 48, 53].

However, one aspect that has not yet received due attention when considering

stochastic elements in OP variations is when the agent behavior is uncertain. It is not clear to see that an agent can have stochastic behavior if we think only of robotic agents, which follow a predetermined plan from beginning to end, except in situations where some failure occurs that make it impossible to continue. We can say that the failure of an agent that makes it impossible to continue on the mission is the worst case of stochastic behavior. In this sense, Jorgensen *et al.* proposed the Team Surviving Orienteers Problem (TSOP), where the cost of crossing an edge is replaced by the probability of the agent surviving when crossing the edge [34]. The goal of TSOP is to find a set of routes, one for each agent on the team, in order to maximize the expected number of nodes collectively visited by the team and maintain the probability of reaching the endpoint for each agent above a specified survival limit. The authors also present some variants and applications such as edge reward and patrolling, multiple visits, and IPP. An approximate greedy algorithm was developed to generate the paths, and extensions to the variants were discussed. The effectiveness of the algorithm was demonstrated in numerical simulations.

Jorgensen *et al.* extended the previous work by presenting an online update version of the problem and a version considering heterogeneous teams [35]. In the online version, at each iteration in the mission execution, information on the probabilities of the paths is updated according to what has already occurred in the mission. When failures in an agent are perceived, the routes are re-planned considering the new probabilities and the surviving agents on the mission. In addition to maximizing the expected cumulative reward, this re-plan aims to distribute the risk fairly to the surviving agents to finish the mission. The problem has also been extended, both static and online, to a heterogeneous environment with different types of agents. The main difference is that the functions and variables that previously depended only on the path now rely on the path and robot type. For all the proposed variations and applications, the necessary adjustments to the algorithm were discussed to run correctly. The algorithm's effectiveness was demonstrated in numerical experiments for various configurations. The authors showed the performance gains from the execution of the online version and using heterogeneous teams.

A work that considers the possibility of failures of the agents already in the planning and that deserves great attention is the work of Shi *et al.* [57]. They proposed Robust Multiple-Path Orienteering Problem (RMOP), which aims to create redundancy in the plans of the agents in order that, even if some agents fail, the team's overall performance will not drop significantly. As the authors report, the worst case on a mission is when all team agents fail. Therefore, RMOP treats the situations where a certain number of α agents, with α less than the number of agents in the team, can fail. The proposed algorithm makes use of the OP to calculate routes individually for each agent. The routes are ordered by the final reward they would generate and separated into two subsets, S_1 with the α most valuable and S_2 with the rest of the routes. The routes in S_1 can overlap with each other and with the routes in S_2, to introduce redundancy to the team. The

routes in S_2 are recalculated using an OP instance so that they do not overlap with each other, to add coverage to the environment. As a reward of a node is collected only on its first visit, the challenge is to resolve the trade-off between overlap and coverage in order that the redundancy of the paths does not penalize the team's reward collection. The results show that the proposed algorithm excels at baselines as the number of agents that fail increases. For cases where there are no failures, the algorithm performs worse due to the budget spent, guaranteeing the redundancy of visits.

2.3 Regions with different importances

When we have to inspect some areas, such as agricultural monitoring or search and rescue, there are regions in the environment that are more valuable to be sensed than others. The critical areas of search and rescue missions are those with potential survivors. In crop monitoring, for example, the most important regions can be those under some treatment or those close to areas with recent outbreaks of pests or diseases. In the literature, we have some variations of the OP that we can approximate to this situation: the ones that use mandatory nodes in the agents' route.

Lin and Yu proposed the Team Orienteering Problem with Time Windows and Mandatory Visits (TOPTW-MV), where some nodes are important nodes that must be visited while others are called optional nodes [45]. The TOPTW-MV objective is to maximize the routes' reward, respecting the time window of each node service and mandatorily visiting the important nodes. The authors formulated a Mixed Integer Linear Programming (MILP) model and developed an Multi-start Simulated Annealing (MSA) heuristic to solve the problem. They compared the MSA performance with the MILP model in 72 small instances, with 25 nodes and mandatory visits ranging from two to five. The solver used to solve the MILP model ended the execution after 30 minutes or when found an optimal solution. The MILP solver returned an optimal solution for 41 instances, also found by the MSA heuristic. Besides, the MSA method produced better solutions than the MILP solver in 13 other instances. Another advantage of the MSA method was the average time to resolve small instances, which was about 4.0 seconds, while the MILP solver needed much more time. Another test was carried out on 168 larger instances, with 100 nodes and five mandatory visits. The performance of the MSA method was compared with an Simulated Annealing (SA) heuristic and an artificial bee colony. The performance of MSA was better than the other two methods regarding the average gap for the best solution value known for each instance.

In the same stream of thought, Assunção and Mateus proposed the Steiner Team

Orienteering Problem (STOP), a generalization of TOP that considers a subset of nodes that must be compulsorily visited [8]. The authors proposed a cutting-plane algorithm to solve the problem exactly. To evaluate the algorithm's performance, they used as a baseline an adaptation of the state-of-the-art algorithm for TOP, proposed by Bianchessi *et al.* [14]. Both algorithms were set to end with a time limit of two hours of computation. The first test verified the proposed algorithm's behavior by solving a TOP benchmark without considering mandatory nodes. They claim that the cutting-plane algorithm solved 15 more instances than any other exact algorithm in the literature about TOP. Also, their approach found eight optimal solutions for instances that had not yet been resolved. The second experiment was specifically for STOP, using TOP instances adapted to the problem with mandatory nodes. The proposed algorithm solved 30 more instances than the defined baseline.

After that, Assunção and Mateus proved that finding a feasible solution to STOP is NP-hard and proposed an Large Neighborhood Search (LNS) heuristic for the problem [9]. They use the formulation presented in the previous work and use it with the Feasibility Pump framework to find a feasible initial solution. The proposed LNS algorithm improves the quality of the initial solutions using classical local search procedures widely used in general routing problems. The proposed heuristic also uses a long-term memory component based on Path Relinking. The results obtained in the experiments were compared with the results of the cutting-plane algorithm proposed in the previous work. On average, the heuristic had a quality gap of 0.54% compared to baseline bounds, showing its efficiency and effectiveness.

In monitoring missions over large areas, the mandatory visit to a node may not be feasible. However, we can consider the existence of valuable nodes whose visit is quasi-mandatory for a good solution to be obtained. Furthermore, in instances with reasonable amounts of nodes, the valuable ones will probably not appear isolated; next to a valuable node, we will have more valuable nodes. The Clustered Orienteering Problem (CluOP) can approach this idea of regions with more valuable nodes than others. Angelelli *et al.* proposed this generalization based on the OP, where the nodes are grouped into subsets called clusters [4]. Each cluster has a profit that is received just if all nodes belonging to the cluster are visited. An exact branch-and-cut algorithm and a Tabu Search heuristic were proposed to solve the problem. The authors believe the problem is more challenging to solve than the OP, especially in calculating an optimal solution. While works in the literature can optimally resolve the OP for instances with up to 500 nodes, the authors report that they could not resolve instances with more than 318 nodes.

A while later, Yahiaoui *et al.* proposed the Clustered Team Orienteering Problem (CluTOP), a variant of CluOP with multiple agents [74]. The authors proposed an algorithm based on a cutting planes approach that includes a pre-processing procedure that computes incompatibilities between clusters and reduces the burden of the exact method.

They also proposed a hybrid heuristic that combines an Adaptive Large Neighborhood Search (ALNS) and a split procedure. The split procedure extracts solutions with better profit on the giant tours generated by ALNS. The authors performed tests on the algorithms considering a single agent to be able to compare with the results of Angelelli *et al.* [4]. The proposed methods have outperformed the literature, with the exact method solving 61 more instances, while the proposed heuristic found 38 new best-known solutions. Regarding the case of multiple agents, the heuristic method proved to be efficient in finding the optimal solution for almost all instances resolved by the exact method.

2.4 Teamwork and heterogeneous agents

The cooperation between agents with different characteristics in the same mission is an increasingly studied subject. Van Der Merwe *et al.* proposed the Cooperative Orienteering Problem with Time Windows (COPTW) and used the term cooperative to indicate that a task needs to be performed cooperatively, in other words, that multiple agents must visit the same point at the same time and within a time window so that the reward is collected [68]. The authors proposed a two-index vehicle flow formulation for the problem and performed tests to indicate the size of the instances that could be resolved in a reasonable time. The results showed that an optimal solution for most problems with 20 nodes or less could be found. For larger instances of the problem, the authors say that the time taken to solve the problem depends greatly on the problem's parameters. Increasing the duration of the time window or the number of vehicles increases the solution time.

In a resource-constrained continuous surveillance scenario, Reardon *et al.* studied the COP to model the solution, but including a human agent in the planning phase [51, 52]. While the autonomous agent is responsible for exploring the environment to maximize the probability of detecting a target, the human agent collaborates with all its expertise in planning the autonomous agent's route. The tour route is planned by solving an instance of the COP, where the rewards are determined by a theoretical information function to optimize the potentiality of detecting targets within the available budget. The human interaction in this planning occurs by specifying or updating the probability of finding a target in some location, shaping the information function used for decision-making. The robot can recalculate the route from the human's tips to adapt to human beliefs, but without any specific order for this. The authors performed experiments in simulated and real-world environments, and the results showed that with a small number of interactions, it was possible to leverage human knowledge to create more successful surveillance routes. It is worth mentioning that the human-robot interaction occurs in the route planning

phase and that it does not change during the execution of the surveillance mission.

An approach that explicitly considers heterogeneous agents involved in a sensing mission is presented by Tokekar *et al.* [62, 61]. They proposed a system where a Unmanned Aerial Vehicle (UAV) and a Unmanned Ground Vehicle (UGV) cooperate in a way that the UGV can mule the UAV between two deployment locations such that the UAV does not spend any energy. However, in this approach, only the nodes to be visited by the UAV are calculated using the OP. To calculate the UGV route, which also makes soil measurements, they proposed a TSP variation called Sampling Traveling Salesperson Problem with Neighborhoods (S-TSPN). In an experiment, the authors show that monitoring done with only an UAV was restricted to a region of the environment. In contrast, the system that combines UAV and UGV can cover more spread and distant points, increasing the information captured in the environment.

Sakamoto *et al.* proposed the Generalized Team Orienteering Problem (GTOP), whose concept is illustrated in Figure 2.3, to incorporate the heterogeneity of agents in an exploration mission [55]. The main difference from GTOP to TOP is a set of abilities present both in the agents and as requirements for executing tasks spread throughout the environment. An agent has more than one ability, and tasks offer a reward defined according to each type of agent's abilities. Thus, the reward collected for performing each task depends on the type of agent who visits that location. The authors proposed an algorithm based on Self Organizing Map (SOM) that showed solutions with better quality than a greedy algorithm proposed. Also, the reported time complexity in the system is constant in the number and types of agents. This complexity favored an exciting experiment, which consisted of applying the algorithm in a moon cave exploration scenario, where the information appears as the exploration proceeds. In these scenarios, it is necessary to repeat the planning to incorporate the new information into the environment. In the results, it was qualitatively shown that the proposed algorithm explores a larger area than the greedy strategy. Besides, the proposed algorithm has a significant advantage in scalability to the number of agents and heterogeneity of the team.

2.5 Two-stage approaches

A monitoring mission can be divided into two stages: planning and execution. Most approaches focus on the first stage of the problem without considering what may happen during the execution stage. Other ones try to minimize problems that may occur in the execution of the mission by adding stochastic elements in the planning stage, as happens with the works described in Section 2.2. We are interested in an approach where

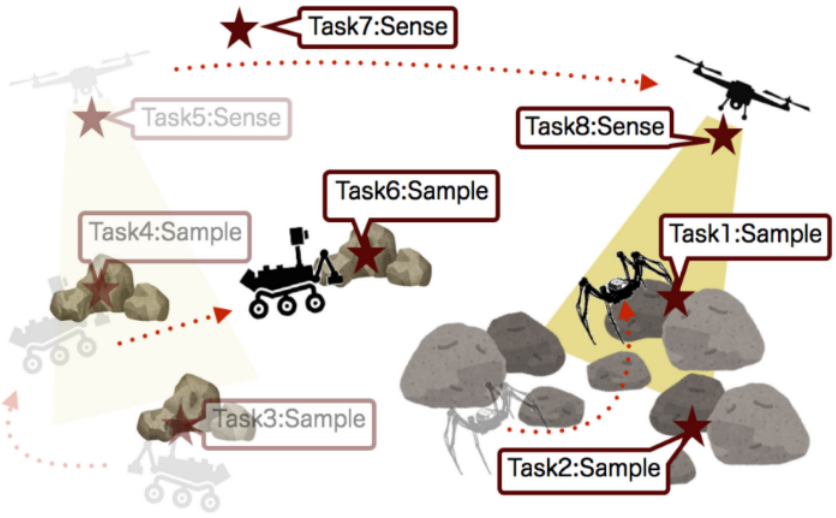


Figure 2.3: Concept of GTOP in an exploration scenario. There are several observation tasks that require different types of sensors or mobility distributed in the fields. Transparent tasks described in the figure represent tasks already executed by robots [55].

the state of the world is continuously updated, and replanning methods are performed online to adjust routes. A work that does not use the two-phase approach, but it is important to understand other works discussed here, is the Orienteering Problem with Stochastic Travel and Service Times (OPSTS). In the OPSTS, Campbell *et al.* considered the travel times between one node and another and the service time at each node as stochastic elements [16]. The route planning considers known probability distributions, and travel and service times are discovered only during the mission execution. Due to these stochastic values, not all nodes belonging to the planned route can be visited, respecting the mission time budget. Therefore, besides the reward, each node has a penalty to be applied if it belongs to the route but is not visited within the time limit. The goal is to find a route that maximizes the expected mission profit considering the rewards and penalties according to the time limit to finish the mission.

To ensure adequate monitoring of a region considering the stochasticities of the real world, it is interesting to pay attention to the mission execution phase. In this context, Dolinskaya *et al.* presented the Adaptive Orienteering Problem with Stochastic Travel Times (AOPST) [21]. This problem is similar to OPSTS, but besides the nodes that offer rewards and penalties, there are also a set of intermediate nodes called transition nodes. This type of node admitted here is necessary because the nodes are not linked with all the others, thus imposing restrictions on reaching a particular node. The travel times from one node to another are stochastic and only realized after an agent performs the displacement, remaining the same until the mission end. Once that is defined, the authors break down the problem into the planning and execution stages. The planning phase, done before the mission begins, defines the subset of nodes that offer rewards that will be visited and their

visiting order. After that, these nodes and the visiting order remain fixed throughout the execution stage. In the execution phase, the agent can dynamically adapt the traversed paths to reach these nodes in the desired order, according to the information learned about the travel costs between the nodes. These adaptations allow the agent to improve its performance and consequently collect more rewards on nodes within the time limit.

Angelelli *et al.* proposed the Probabilistic Orienteering Problem (POP), where the uncertainty lies in the availability of the nodes to be visited or not [5]. The POP consists of two phases. The first is to plan a route between the start and endpoint, respecting a time limit. In the second phase, information on the availability of the nodes is made accessible, and the mission execution starts visiting the nodes following the generated route in the first phase by ignoring the unavailable nodes. The final mission profit is the difference between the sum of the rewards collected on visited nodes and the route cost, skipping the unavailable nodes. While OP finds a route that maximizes the amount of the collected utility with a budget limit, POP looks for a route that maximizes the expected mission profit for all possible subsets of nodes, also respecting the budget limit.

Evers *et al.* say that in real-life applications, the costs of moving from one point to another in the OP are often uncertain [22]. Therefore, the solution to deterministic OP may be infeasible or sub-optimal. Because of this, they addressed Orienteering Problem with Stochastic Weights (OPSW) with a two-stage strategy. One of the differentials of this work about others that consider stochastic edges is that the authors assume that the capacity constraint on the total route cost is hard. That is, the depot has to be reached before the total realized costs exceed the budget. Thus, the objective in the first phase is to build a route that may have to be aborted in the second phase before reaching the end due to the realization of the edge values. Aborting in the second phase means the agent must return to the depot immediately without visiting the rest of the route nodes. The authors proposed two approaches to solving OPSW: a Sample Average Approximation (SAA) and a heuristic that takes advantage of the problem's structure. SAA is a solution technique widely used in stochastic programming to solve stochastic optimization problems through the Monte Carlo simulation. The resulting SAA problem was a large Mixed Integer Program (MIP), making it impossible for large instances to be resolved within a reasonable time. The proposed heuristic uses a score measure that incorporates a node's reward with the nodes' rewards in its proximity and a local search that aims to find routes with a higher expected profit after the execution phase. The authors reported the performance of several tests, and the results obtained were superior when compared with solutions of the deterministic OP.

A problem that more closely resembles what we are addressing in this dissertation proposal is the Operational-level Stochastic Orienteering Problem (OSOP), proposed by Bian and Liu [13]. The proposition considers uncertain factors on the operational level of the OPSTS, dynamically revealed after an agent visits certain points of interest, possibly

causing the a priori planning to be no longer optimal. The authors present a real-time strategy to adjust the route planning adopted by the agent, which acts lonely in the mission. The adjustment procedure occurs after the agent finishes a point of interest visit due to the travel time to this point, and the service time spent on it is discovered only at that moment. For example, after visiting a point of interest, the agent may have time to visit more locations or be unable to visit all the points on the route with the remaining time budget. The course is adjusted to collect more rewards and reduce the risk of not reaching the end route point within the time limit.

2.6 Problem contextualization

As presented in this Chapter, many variants exist for the OP. We describe some of these variants with characteristics that we consider essential for the problem addressed in this dissertation. We are interested in a relatively recent variant known as CTOP. The CTOP alone already represents a significant challenge because, in addition to creating routes limited by a budget for each team member, it uses a utility function that depends on both the visited and not visited nodes. We seek to contribute to each of the points highlighted in this Chapter's sections and propose methods that consider all of them together in a CTOP instance, further increasing the degree of difficulty of the problem.

Table 2.1 summarizes the main works related to the problem found in the literature, highlighting which specific features they address and contextualizing the problem addressed in this dissertation. The *Correlated* column indicates if it is considered a correlated reward between nodes, while the *Stochasticity* one presents the elements considered stochastic. We consider problems that foresee agents' failures as stochastic in the agent's behavior. If the approach considers single or multiple agents and whether the team is homogeneous or heterogeneous is reported in the *Team type* column. If the problem has mandatory nodes or regions of valuable nodes that will necessarily be in an optimal solution, we indicate in the (quasi-)mandatory column (*(q)mandat.*). Finally, in the *Online* column, we inform if the method changes the routes during the mission execution phase.

Table 2.1: Main works found in the literature and the characteristics addressed by them, contextualizing the problem proposed in this dissertation proposal.

	Correlation	**Stochasticity**	**Team type**	**(q)mandat.**	**Online**
Yu *et al.* [75, 76]	✗		single homogeneous		
Tsiogkas *et al.* [64, 65]	✗		single		
Arora and Scherer[7]	✗		single		
Tsiogkas and Lane[66]	✗		homogeneous		
Tsiogkas and Lane[67]	✗		single		
Jorgensen *et al.* [34]		edges	homogeneous		
Jorgensen *et al.* [35]		edges	heterogeneous		✗
Shi *et al.* [57]		agents' behavior	homogeneous		
Kobeaga *et al.* [39]			single		
Santini *et al.* [56]			single		
Kobeaga *et al.* [40]			single		
Hammami *et al.* [29]			homogeneous		
Lin and Yu [45]			homogeneous	✗	
Assunção and Mateus [8, 9]			homogeneous	✗	
Angelelli *et al.* [4]			single	✗	
Yahiaoui *et al.* [74]			homogeneous	✗	
Van Der Merwe *et al.* [68]			heterogeneous		
Sakamoto *et al.* [55]			heterogeneous		✗
Campbell *et al.* [16]		travel and service times	single		
Dolinskaya *et al.* [21]		travel times	single		✗
Angelelli *et al.* [5]		nodes' availability	single		✗
Evers *et al.* [22]		travel times	single		✗
Bian and Liu [13]		travel and service times	single		✗
This dissertation	✗	agents' behavior	heterogeneous	✗	✗

Chapter 3

Methodology

Most literature work treats the Orienteering Problem (OP) and its variants only by creating a static plan followed by agents in an area monitoring mission. In addition to most often not reflecting the individualities of agents concerning locomotion costs and perception capability, these works also do not consider events that occur during mission execution that might prevent an agent from doing what is in the mission plan. These events include mobility issues, such as unforeseen roadblocks in the planning phase, defects in the sensors that capture information from the environment, or any other physical or logical failure that makes it impossible for the agent to accomplish the plan.

When considering that an initial plan for an area monitoring mission was calculated optimally or close to it when using a heuristic approach, we can say that the occurrence of failures in the agents during the mission will harm the final collection of team rewards as initially planned. In these cases, it would be interesting to try to recover part of the reward that the failed agents would no longer capture so that these failures don't have such a big impact on the team's total prize at the end of the mission. We then propose a strategy to adapt the routes of active agents in the mission, regardless of how the initial plan the monitoring team would follow was generated. In addition, we propose a route planning methodology that considers the possibility of failure of one or more agents during the plan execution phase and generates routes that seek to minimize the impact of losses and benefit the adaptation strategy when necessary.

For this purpose, we describe in Section 3.1 how we represent the environment and the characteristics of each point of interest. Section 3.2 formally defines the problem attacked in this dissertation, considering generic teams of agents. To better organize the narrative, we divide our approach into two chapters according to the team type, describing the methodology and presenting the results obtained with each one. Chapter 4 considers teams composed of agents with equal capabilities, like most works that address the CTOP. In Chapter 5, we investigated the impact of using teams formed by agents with different budgets and perception capabilities.

3.1 Environment representation

One of the first concerns when conducting a monitoring mission in an environment is how we represent that environment. A widely used approach is describing the environment by a finite set of Points Of Interest (POIs), as illustrated in Figure 3.1. Each POI is equivalent to a sensing location where the agent must physically visit to capture environmental information using its sensors. Figure 3.1a illustrates a projection of POIs over an area to be monitored, while in Figure 3.1b, we represent an environment by a regular grid of POIs.



(a) (b)

Figure 3.1: Example of discrete environments using POIs. (a) set of POIs depicting a rural area, projected over it; and (b) squared area represented by a 5×5 grid of POIs.

Each POI rewards the system when an agent visits it. This reward is measured by a utility function defined according to the purpose of the monitoring mission and the agent's capability to perceive. Moreover, particularly in this dissertation, the utility function for a visited POI is also influenced by the information in nearby unvisited locations, as indicated in Figure 3.2. We can use functions like this in environments with a correlation between information from adjacent regions, common in the area monitoring task. We define a correlation range to restrict which points influence the utility function of a visited POI. The correlation range can differ from agent to agent and suffer interference from environmental noise.

Figure 3.2 illustrates the correlation range of a visited POI. The large dotted circles at the sides of the figure represent in detail the correlation range of 1.5 distance units when an agent visits the red dot in the center of the grid. POI labels represent the rewards the agent collects when visiting that point. The dotted lines, which connect the red POI to the others, have labels that indicate the correlated reward (c_{rwd}) between the points or the distance between the locations. The circle on the right side represents the distance between the central POI and the others within the correlation range. The left side circle represents the correlated reward that the agent collects when visiting the red POI when the correlated point is not visited. This value is calculated using a function that considers

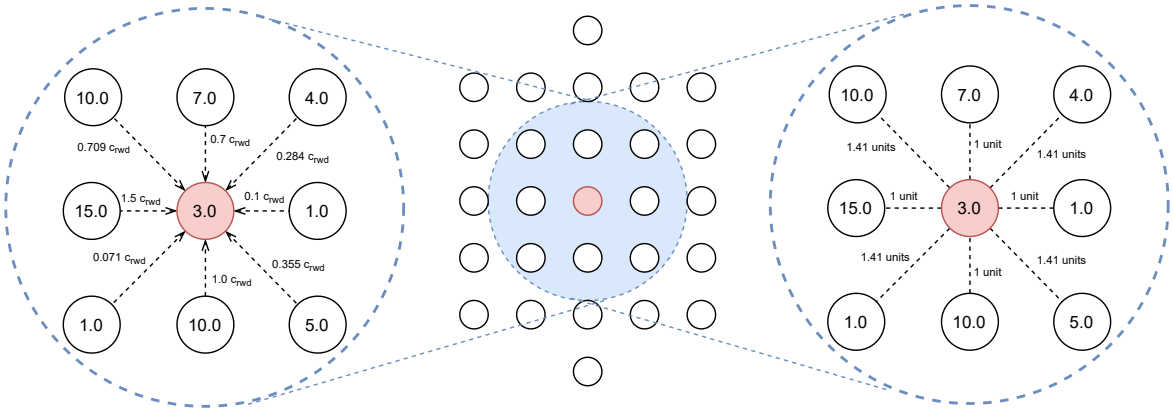


Figure 3.2: Reward and the correlation range of the red POI at the center of the grid that are considered in the utility function's calculation. The big dotted circles on the sides show in detail the POIs within the correlation range of the red POI. The left circle shows an example of the reward gain the red POI would receive from its neighbors, using the example function $c_{rwd} = \frac{0.1}{dist(\cdot)} \times reward$, while the right one shows the distance values.

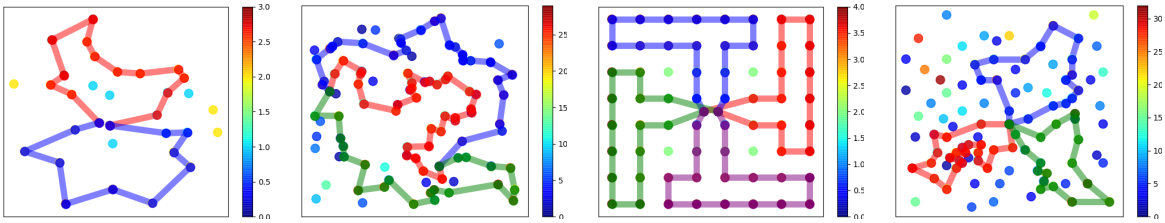
the POI reward and its distance from the visited POI. In the example of the figure, we use the function $c_{rwd} = \frac{0.1}{dist(\cdot)} \times reward$ to illustrate.

Since we focus on monitoring areas with regions that are more important than others to inspect, and the CTOP instances available in the literature did not have this characteristic, we create instances that represent this scenario. The next section describes the set of instances used in the experiments and shows some examples.

3.1.1 Instance set

As far as we know, there is no benchmark with instances for the CTOP that we could use in our experiments. Thus, we made preliminary tests of our proposal using instances for TOP. As the difference from TOP to CTOP is the utility function, to transform a CTOP instance into a TOP instance, use the correlation range equal to zero. In this way, the neighbors' reward does not affect the utility function of a visited node. Considering the correlation range equal to zero, we opted for the benchmark provided by Chao *et al.* [19], which is widely used for experiments on approaches for TOP and, for this reason, most instances of this benchmark have a known optimal solution. Our goal is to test our method's behavior during the mission, so we use these known solutions as a starting point for our sensing mission.

The experiments' results in these instances made us understand that these benchmarks would not be the best option for what we were planning. Figure 3.3 shows some of these instances with an optimal route configuration.



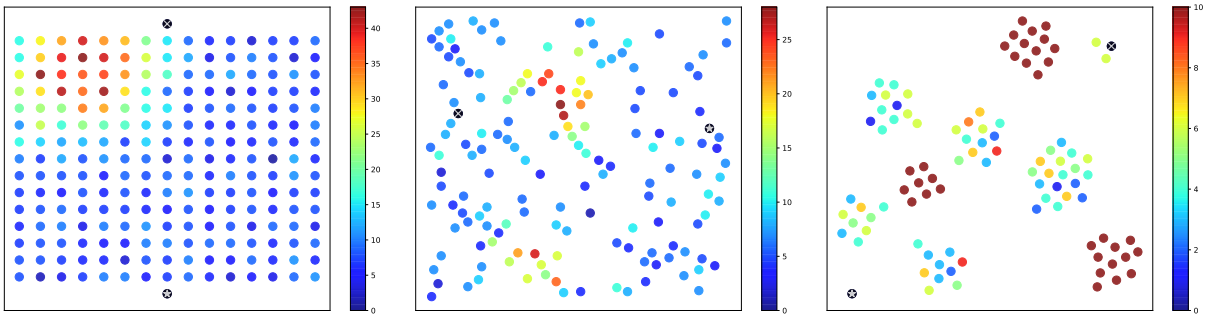
(a) p1.2.n (b) p4.3.p (c) p5.4.z (d) p7.3.r

Figure 3.3: Example of instances of the benchmark proposed by Chao *et al.* [19]. The routes shown are optimal for each instance according to the number of agents and the available budget. The images are not at the same scale.

The instances of this benchmark were designed for a deterministic approach to the problem. Thus, the optimal routes are well dispersed among the environment, and the agents use nearly all of their budget to maximize reward collection. In our case, as we considered the failure of an agent during the mission, the recovery of reward in these instances would not be feasible. For example, in Figure 3.3a, if one agent fails, it would not be worth adjusting the other agent's route to recover part of the reward that would be lost, as the two paths are opposite in the environment. In Figures 3.3b to 3.3d, it may seem advantageous to be adapting agents' routes in the event of a team member's failure. However, in practice, this is not true. For an agent to reach a node that a teammate would no longer visit, it would have to remove many nodes from its route to increase its budget. But, in this case, removing a large number of nodes to reach just one node would be a bad choice. Firstly, the agents' routes shown in the figures are optimal, and besides that, there are no nodes much more valuable than others that be worthwhile to make changes to the routes. In other words, no regions in the environment are more profitable to visit than others, as are the cases we are addressing in this dissertation.

To have scenarios that best meet our initial objective of monitoring areas with regions more important to inspect than others, we propose our own set of instances. In all instances, each node has a designated reward given by a normal distribution $\mathcal{N}(\mu, \sigma^2)$, with $\mu = 10, \sigma = 2$. An exception is the nodes indicated as the beginning and end of the routes, which do not reward the system for being visited. Besides, we want to consider some regions that have priority to be visited. It is the case of areas with the possibility of explosion or collapse in search and rescue missions and plots under treatment or with a tendency to be attacked by pests in agricultural monitoring missions. To represent this feature in our instances, we manually selected some POIs according to the number of desired regions. We increased 4× the reward offered by all nodes within a radius of 2 meters from the selected points concerning that generated with the normal distribution. The intention is that these regions were much more valuable than the others and that agents must pass through them in the mission execution.

We distribute the POIs across the environment in three ways to make the instances



(a) (b) (c)

Figure 3.4: Examples of generated instances. The rewards offered by each point follow the colors of the bar placed next to each instance. Small regions with a reddish color indicate the regions more important to inspect. The images are not at the same scale.

more embracing. Figure 3.4a shows one example of an instance in a grid format. In these instances, the POIs were equally spaced with a vertical and horizontal distance of 1 meter between them. The start and end points of the routes were positioned centrally below and above the grid, respectively. The instance shown in Figure 3.4a has one more important area to inspect, which appears in the upper left part of the image. We also created instances by scattering the POIs randomly across the environment without a predefined organization, as illustrated in Figure 3.4b. We defined the area size according to the number of points: we calculated the necessary size to place the points in a grid format and then duplicated this area to spread the POIs. Points can be placed anywhere in this defined area, including the start and end points of routes. The only rule used is that the POIs could not overlap. Figure 3.4b shows an instance created along these lines. The starting point is the one with a white star inside, and an x indicates the arrival point. Furthermore, this instance has two regions with more valuable points than the others. We create a third instance group that groups the POIs into clusters, as shown in Figure 3.4c. We define the size of the area in the same way as we did for the instances with random positioning. After that, we manually position the POIs in that area, taking care to create the clusters of points.

3.2 Problem Formulation

We are interested in the problem of monitoring areas with different importance considering failure-prone agents. An overview of this problem is illustrated in Figure 3.5. Given a set of POIs and a team of agents, our main goal is to provide ways to reduce the impact of team agent failures during mission execution.

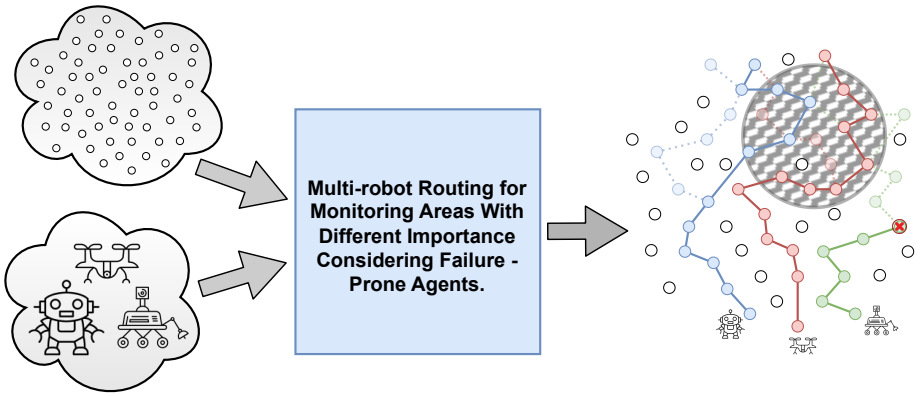


Figure 3.5: Overview of the problem addressed in this dissertation. The agent following the green route failed, and the others adapted their routes to cover the most important region indicated by the hatched area.

This Section describes how we model this problem considering a team of generic agents, i.e., initially, we are not concerned if the team agents have the same or different capabilities. That said, we modeled the problem as an instance of Correlated Team Orienteering Problem (CTOP) [75, 76]. In this sense, let $\boldsymbol{\mathcal{V}} = \{v_1, \ldots, v_n\}$ be a set of n POIs in some workspace $\mathcal{W} \subset \mathbb{R}^2$. Each POI $v_i \in \boldsymbol{\mathcal{V}}$ is characterized by its spatial location $o_i \in \mathbb{R}^2$ and a utility (reward) $r_i \in \mathbb{R}^+$ offered by being visited. Also, be $v_{orig}, v_{dest} \notin \boldsymbol{\mathcal{V}}$ two particular points that do not offer rewards and serve as origin and destination locations for all agents, respectively. Let $\boldsymbol{\mathcal{A}} = \{a_1, \ldots, a_m\}$ be a team composed of m agents, where each agent can have its particularities, but each must have its budget available for the execution of the mission. We define a route π as an ordered subset of POIs from the start to the goal position, i.e., $\pi = \langle v_0^\pi, v_1^\pi, v_2^\pi, \ldots, v_{|\pi|-2}^\pi, v_{|\pi|-1}^\pi \rangle$, where $v_0^\pi = v_{orig}$ and $v_{|\pi|-1}^\pi = v_{dest}$. A route segment is a continuous portion of π defined as $\pi^{[b:e]} = \langle v_b^\pi, \ldots, v_e^\pi \rangle$, with $0 \leq b, e < |\pi|$ and $b \leq e$.

The CTOP aims to generate a collection of routes, one for each team member, respecting each agent's budget constraints, in order to maximize the sum of utility gathered by visiting, for the first time, POIs belonging to $\boldsymbol{\mathcal{V}}$. This implies multiple visits to the same POI do not contribute to the collected reward. In each visited node v_i, extra utility is gathered from spatially correlated nodes in v_i's neighborhood. This neighborhood is bounded by a correlation range (cr) and defined as the set $\boldsymbol{\mathcal{N}}_{v_i}^{cr} = \{v_j \mid dist(v_i, v_j) \leq cr\}$, where $dist(v_i, v_j)$ is the euclidean distance between v_i and v_j in $\mathbb{R}^2$. The extra utility from $\boldsymbol{\mathcal{N}}_{v_i}^{cr}$ is the sum of each unvisited neighbor v_j multiplied by weight a w_{ij}^k, which indicates the amount of information v_i can get from its correlation with v_j. The total utility gathered at each visited node v_i is given by

$$\textsc{Reward}(v_i) = r_i x_i + \sum_{v_j \in \boldsymbol{\mathcal{N}}_{v_i}^{cr}} r_j w_{ij} x_i (x_i - x_j), \tag{3.1}$$

where x_i is a variable that assumes the value 1 when node v_i is visited in one of the solution routes or the value 0 otherwise. When v_i is not visited, the two terms of Equation 3.1 become zero because both are multiplied by $x_i = 0$. In contrast, in addition to the utility to visiting v_i, given by the first term of the Equation 3.1, an extra utility correlated with its neighbors v_j is also computed by the summation term. Only unvisited nodes would be considered at the summation using the quadratic term $x_i(x_i - x_j)$, which will be nonzero when $x_i = 1$ and $x_j = 0$, that is, when v_i is visited, and its neighbor v_j is not. The overall utility gathered by an agent considering the route segment $\pi^{[b:e]}$ is computed as:

$$\textsc{RouteReward}(\pi, b, e) = \sum_{i=b}^{e} \textsc{Reward}(v_i^{\pi}), \tag{3.2}$$

where v_i^{π} is the i-th element in π. To compute the utility of the entire route π, we use $b = 1$ and $e = |\pi| - 2$. Note that we do not consider v_{orig} and v_{dest} because they have no rewards and serve only to indicate the beginning and end of the route.

Besides the utility of a route, each agent a_k has it's budget β_k comprising the energy, distance, or time restrictions. Let $\mathcal{C}(v_i)$ be the cost of collecting a utility when visiting the node v_i, and $\mathcal{D}(v_i, v_j)$ the cost of visiting v_j immediately after visiting node v_i, a segment route cost is given by

$$\textsc{Cost}(\pi_k, b, e) = \sum_{i=b}^{e} \mathcal{C}(v_i^{\pi_k}) + \sum_{i=b}^{e-1} \mathcal{D}(v_i^{\pi_k}, v_{i+1}^{\pi_k}). \tag{3.3}$$

Thus, in the cost of the entire route π, we consider $v_{orig} = v_0^{\pi}$ and $v_{dest} = v_{|\pi|-1}^{\pi}$, and use $b = 0$ and $e = |\pi| - 1$. Note that an agent can pass through a point and not collect the utility from it, in which case the collection cost is not charged, and the utility from that point is also not computed for the team. In this dissertation, we consider that the budget and all the costs involved in the mission are related to the time to perform some task.

Finally, the total utility to be maximized is the sum of the collected rewards from visited nodes and the correlated rewards from its neighbors. The CTOP objective is to find a set of routes $\mathbf{\Phi} = \{\pi_1, \ldots, \pi_m\}$ in the space of all possible routes Ψ in order to find the maximum utility, i.e.,

$$\begin{aligned} \mathbf{\Phi} = \arg\max_{\mathbf{\Phi} \in \Psi} \sum_{\pi \in \mathbf{\Phi}} & \textsc{RouteReward}(\pi, 1, |\pi| - 2), \\ \text{s.t. } & \textsc{Cost}(\pi, 0, |\pi| - 1) \leq \beta_k \\ & \forall k \in \{1, 2, \ldots, m\}. \end{aligned} \tag{3.4}$$

Up to this point, we have described how to obtain the planning of a monitoring mission for a team of generic agents using the CTOP. However, we are also interested in a step beyond planning. Thus, we will address the problem of monitoring areas using CTOP

in two phases, as illustrated in Figure 3.6. The first phase consists of creating a plan for a team with failure-prone agents, following what was described in this Section, in order to maximize Equation 3.4. The second phase concerns the execution of the planned mission. In this phase, we consider that unforeseen events may prevent an agent from fulfilling his part of the mission. Therefore, we have to provide some method for recovering part of the utility that the team would no longer collect so that the assessment of Equation 3.4 at the end of the mission is not too impaired. We call the problem as *Correlated Team Orienteering Problem with Failure-prone Agents (CTOP-FPA)* and define it in two phases, as follows:

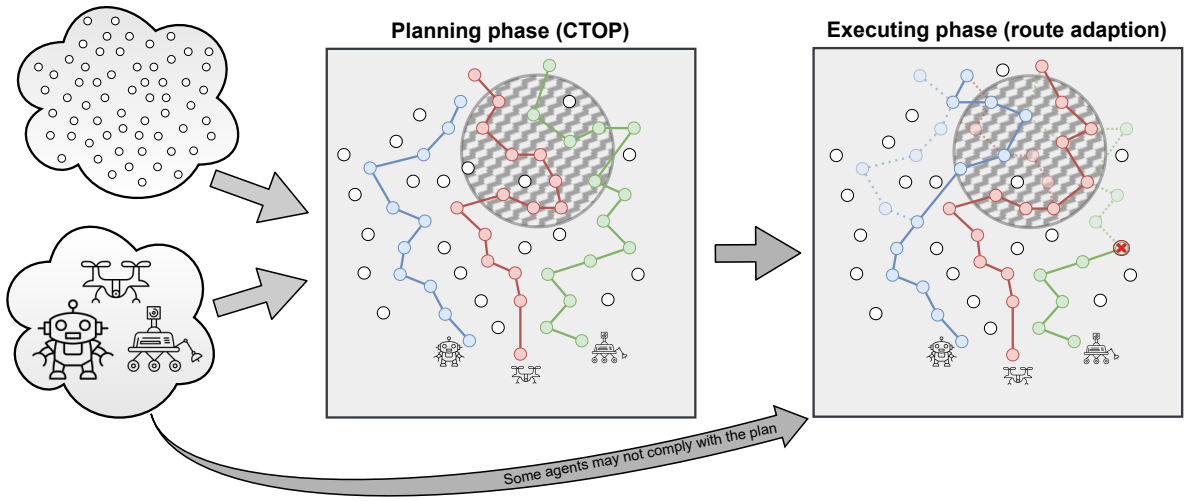


Figure 3.6: Illustration of Correlated Team Orienteering Problem with Failure-prone Agents tackled in two phases.

Problem 1 (Correlated Team Orienteering Problem with Failure-prone Agents)**.** *Given a set $\mathcal{V}$ of points of interest and a set $\mathcal{A}$ of failure-prone agents, each with a budget limit. The goal is to maximize the reward collected by the team, regardless of adverse events that may occur to agents during the execution of the mission. The problem can be divided into two phases:*

***Planning Phase**: Given the sets $\mathcal{V}$ and $\mathcal{A}$, plan a set Φ of routes, one for each agent, respecting the agent's budget, in order to maximize the Equation 3.4 while a possible failure of an agent has a low impact on the evaluation of the Equation 3.4.*

***Execution Phase**: Given the sets $\mathcal{V}$, $\mathcal{A}$, and Φ, perform the monitoring mission. The goal is to minimize the impact of unexpected events in the assessment of Equation 3.4. To that, we need to efficiently adapt the routes whenever one or more team members present failures that prevent it from completing its part of the mission, however, still complying with each agent's budget constraint.*

We address this problem considering two different team configurations and detail these approaches in Chapters 4 and 5. In Chapter 4, we consider teams with agents

with equal capabilities. We conducted this study because most of the works on CTOP in literature use this type of team, and thus we could position our strategy against the existing ones. In addition, another factor that motivated us to perform this approach was investigating whether a route adaptation strategy in the execution phase would benefit the team in cases where one or more agents fail. In Chapter 5, we extrapolate the idea to teams of agents with different capabilities. We believe that heterogeneous teams better approximate real-world situations and that problem modeling focused on this type of team would greatly contribute to the area.

Chapter 4

Homogeneous teams

In this Chapter, we address the problem using teams with homogeneous agents; that is, all agents have the same budget, equal costs to perform each task and likewise perceive the environment. This approach is important to understand if it is worth considering possible agent failures in the planning stage and if, in case of failure of any agent in the mission execution stage, an adaptation in the routes of agents still active would be more cost-effective than a total mission redesign. To better understand what has been developed, this Chapter is divided into sections that follow the chronological order of the development of our approach. Section 4.1 describes the initial planning method we used when beginning to investigate this problem, while Section 4.2 describes the route adaptation routine we propose for the execution phase. Section 4.3 shows a planning method that considers the possible failure of one or more agents and benefits the suggested adaptation routine. Finally, Section 4.4 presents the concluding remarks about this discussion.

4.1 Planning Phase

The planning phase corresponds to the initial route generation for a monitoring mission using an instance of Correlated Team Orienteering Problem (CTOP). Since initially we did not propose any changes to the problem, the first attempt was to use the MIQP method proposed by Yu *et al.* [76] to calculate the team routes. However, when the instances have significant POIs amounts, the MIQP method has become unfeasible to return good solutions in a reasonable computation time. To overcome this issue, Tsiogkas and Lane proposed a heuristic to solve CTOP instances faster than the MIQP method and with promising results [66]. When we started this investigation, it was one of the few heuristics in the literature specifically designed for the CTOP and is still considered state-of-the-art in heuristically solving CTOP. The authors also reported that the method had the potential to redesign online routes in the event of a change in the mission parameters, which would fit the agent failure cases we are interested in.

The Tsiogkas and Lane heuristic is a Genetic Algorithm (GA) where a single route is equivalent to a gene, and a set of genes represents a chromosome. Thus, we can say that a chromosome is a team plan or a solution for an CTOP instance. Besides, in the proposed GA, a population is a predefined amount of chromosomes that will serve as a basis for the optimization step. To create the initial population, they progressively build a route by inserting one of the free neighbors from the last POI added to the route. This neighbor is chosen based on the distance from all other POIs into the route and the number of free neighbors. This process is repeated until it is no longer possible to add POIs to the route due to a lack of budget.

The optimization phase repeatedly applies the selection, crossover, and mutation operations. For each iteration of the optimization process, we name a generation. The population is renewed between generations, seeking to produce better chromosomes. In the crossover operation, they randomly choose two chromosomes to give rise to two new ones. Again randomly, one of the source chromosomes is selected to provide its best gene (route) to be inserted into the chromosome being created. Another process applied between generations is mutation by adding a POI to the gene or removing one. In the removal case, the removed POI is the least contributor to the gene's fitness. In adding a POI, they choose a random free node and insert it at the best place found in the gene, or if the route cost is 95% of the maximum cost, they swap a random POI of the gene with one of its free neighbors. The POI is only inserted if it does not violate the maximum cost constraint.

The authors compared the heuristic performance with the optimal MIQP solution based on the reward captured by the team and the time spent generating the plan. The heuristic generally performed at least as well as 5% below the optimal reward. Regarding solution time complexity, the heuristic performs at least 300 times faster than the MIQP method. The authors emphasized that these results make the method suitable for online use, but the largest instance used contained 81 POIs. Even with reports of experiments only in instances that did not directly reflect the type of area we were interested in, we chose to use this heuristic at the first moment because we were more concerned about the mission's execution phase. We needed initial planning, and the method proposed by Tsiogkas and Lane appeared promising.

4.2 Execution Phase

As we discretize the environment and identify each POI with a unique identifier, a route is nothing more than a sequence of POIs that an agent must visit in the order in

which they appear—remembering that this route must respect the agent's budget. With one initial planning, the objective is to carry out the monitoring mission considering that at some point, one or more agents may fail and be prevented from fulfilling what was planned. Intuitively, after detecting that a team member has failed, we can consider the recurring application of the approach used in the first phase to redesign the mission. Theoretically, this strategy would guarantee better solutions according to the current state of the world.

However, CTOP is an NP-Difficult problem. Thus, applying a method not projected for real-time computing whenever an agent fails is not feasible for executing the mission in areas with a moderate number of POIs and agents. Due to the planning strategies being designed to generate initial routes, reusing them at this point would require that the agents be stationary in the current POI so that a new plan could be calculated. In addition, the time spent on this calculation would often be too high to be used in a real-time application such as the execution of a monitoring mission. To overcome this problem, we present an efficient strategy that makes small adjustments to routes locally, minimizing the global impact of failures.

The reward captured by an agent can be increased by adding POIs to its route. However, only adding new POIs is not always possible due to agent budget constraints. Therefore, adjusting the route involves removing POIs to increase the available budget and inserting new POIs to raise the agent's reward capture. In the route adaptation, we must evaluate Equation 3.4 to take the best actions that optimize the mission's final result. However, the computational complexity of Equation 3.4 increases with the size of $\boldsymbol{\mathcal{V}}$; that is, when $\boldsymbol{\mathcal{V}}$ is large, computing Equation 3.4 might be time-consuming and become impractical in real-time situations. We propose an anytime local greedy strategy to tackle the task's complexity, adjusting only the segment composed of the next la POIs in the agent's route from its current position. The key idea is that the local adaptation will optimize the final gathered reward.

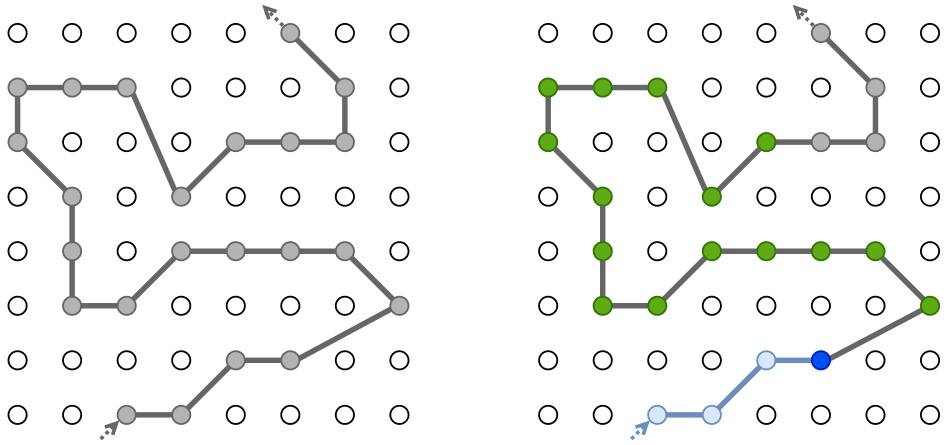
Algorithm 1 shows the steps of our method to adapt routes. The parameters informed for the procedure depend on the agent from which we want to adapt the route. It is worth mentioning that the POIs' current state is shared and known by everyone; therefore, we do not need to pass it to the procedure. The first two input values are the route π and the budget β_k the agent must respect. We also consider the agent's current position pos, the number la of POIs to use in the procedure, and nr, which controls the POIs' neighborhood range used to choose new POIs for the route. The rem parameter indicates the percentage of POIs removed from the analyzed segment to increase the available budget before inserting more valuable POIs into the route.

In Figure 4.1, we see a slice of an instance where we illustrate the concept of some of the algorithm parameters. Figure 4.1a shows a fragment of the route that originates from below the figure, passes through the 23 highlighted POIs, and continues to another

Algorithm 1: RouteAdaptation($\pi, \beta_k, pos, la, nr, rem$)

```
1  u ← EVALUATE(π , pos)
2  π' ← π
3  segment_size ← la
4  repeat
5  |  π' ← REMOVELEASTCONTRIBUTOR(π', pos, la)
6  |  la ← la − 1
7  until remove rem% of nodes from analyzed segment
8  la ← segment_size
9  repeat
10 |  π' ← INSERTFITTEST(π', pos, la, nr, β_k)
11 until can no longer insert nodes in the segment
12 u' ← EVALUATE(π', pos)
13 if u' > u then
14 |  return π'
15 end
16 return π
```

point above. In Figure 4.1b, the dark blue POI represents the current position of the agent (pos). In contrast, the three light blue POIs that precede it on the route are POIs that have already been visited and have already had the reward captured. Following the agent's current position, we have 15 green POIs, representing the parameter la. We work on updating the agent's route on this set of POIs. We will use this scenario described in Figure 4.1b to illustrate other parameters and the operations of the algorithm in the next figures in this Section.



(a) (b)

Figure 4.1: Slice of an instance with a fragment from an agent's route. (a) the highlighted nodes are part of the route fragment and must be visited from the bottom arrow following the indicated line; (b) The dark blue node represents the current position of the agent, while the green nodes represent the segment of the route that will be considered for adaptation ($la = 15$).

Because of the correlated nature of the rewards, a new configuration might be less profitable than the previous one. Therefore, we evaluate before and after the adaptation operations to check whether there is a gain with the POIs readjusting. We must recalculate Equation 3.4 to verify if the changes were worthwhile. However, as stated, this calculation may be infeasible depending on the size of the problem instance. Thus, we apply an alternative strategy to evaluate the agent's route adaptations instead of using all POIs in $\mathcal{V}$ that is present in a route of the solution. The function EVALUATE( ) computes the cost-effectiveness of the agent's remaining route. Assuming only the POIs in the agent's route from its current position, we calculate the reward to be collected according to Equation 3.2 and divide this value by the cost of running that part of the route. The segment is evaluated before and after it is adapted, and we set the most cost-effective one for the agent to keep running.

After calculating the route reward from the agent's current position, we save the route to make changes without losing the original path and maintain the anytime property. Then, we remove $rem\%$ of the POIs present in the analyzed segment. For example, if the analyzed segment has 20 POIs ($la = 20$), and the parameter $rem = 10$, we remove two POIs from the segment. The function REMOVELEASTCONTRIBUTOR( ) removes the POI that least contributes to the solution in the route interval between the current position (pos) and the next la POIs, keeping the path's continuity. The function chooses the POI to remove by solving the following equation:

$$\text{GETLEASTCONTRIBUTOR}(\pi, pos, la) = \min \text{REWARD}(v_i^{\pi}), \quad \forall pos < i < pos + la. \tag{4.1}$$

and we decrease the value of la whenever we do one removal. Thus we ensure that only POIs from the segment indicated via parameter are removed.

We show an example of the removal step in Figure 4.2, considering $la = 15$ and $rem = 20$. With these values, we need to remove 20% of the 15 POIs analyzed, i.e., three POIs of this route segment. The coloring of the POIs follows what we describe in Figure 4.1. While in Figure 4.2a, the yellow POI indicates that one found by solving Equation 4.1, Figure 4.2b shows the route segment after removing that POI. These two steps represent one execution of the function REMOVELEASTCONTRIBUTOR( ). As we have to remove three POIs in this example, this process is repeated twice as long. We can see in Figure 4.2b that the green POIs are now only 14, one less than at the beginning of the process. After calculating Equation 4.1 for this new segment ($la = 14$), Figure 4.2c indicates in yellow which POI contributes the least and should be removed. Figure 4.2d shows the result of removing this POI. Finally, in Figure 4.2e, the yellow POI shows the least contributor in the segment of size $la = 13$, and the final result of the removal step is shown in Figure 4.2f.

After removing some POIs, the agent's available budget increases, allowing the

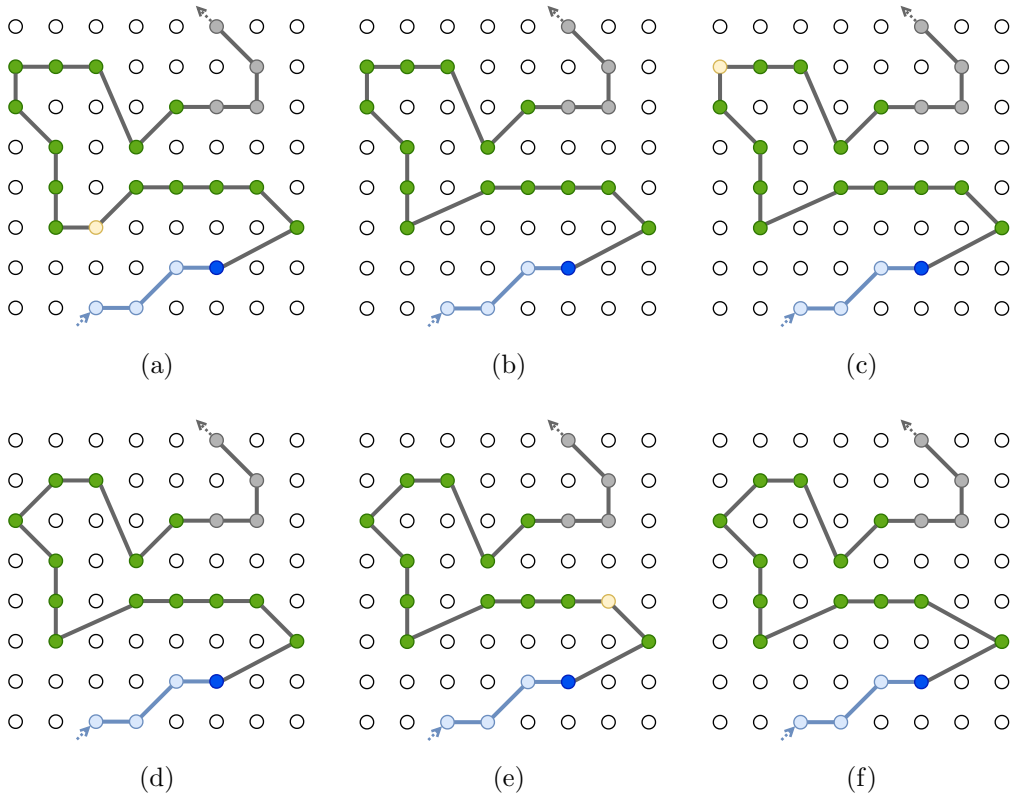


Figure 4.2: Illustration of the three removals performed considering parameters $la = 15$ and $rem = 20$. The green nodes represent the segment of la nodes analyzed, which decreases with each removal, and the yellow nodes are those found by Equation 4.1, and that must be removed.

insertion of new POIs into the route. Thus, the function InsertFittest( ) searches for POIs that are not part of any other path and can be added to the route, respecting the agent's budget constraint. We limit the search space of these POIs using the parameters la and nr. We look only at POIs that do not belong to any route and are in the distance less than nr away from at least one of the la points on the analyzed segment. We select the POI that most increases the reward gain in the newly adapted segment using Equation 4.2:

$$\begin{aligned} \textsc{GetFittest}(\pi, pos, la, nr, \beta_k) = \max(\textsc{Reward}(v_j) \times \mathbb{1}(v_j, \beta_k)), \\ \forall v_j \in \boldsymbol{\mathcal{N}}^{nr}_{v_i^\pi},\ pos < i < pos + la, \end{aligned} \tag{4.2}$$

where $\boldsymbol{\mathcal{N}}^{nr}_{v_i^\pi}$ represents the v_i^π neighborhood bounded by the nr range and $\mathbb{1}(\cdot)$ is an indicator function that returns 1 when v_j does not belong to any agent's route, and we can add it to the π respecting the budget β_k, and 0 otherwise.

Once the POI that maximizes the reward (v_{max}) is defined, we disconnect two POIs from the analyzed segment and reconnect them using v_{max} as an intermediary. The two POIs chosen in the segment are those closest to v_{max} that maintain the route feasibility with the lowest cost. The choice of where the v_{max} will be inserted in the

route is illustrated in Figure 4.3, where we indicate this POI in yellow. The POIs inside the shaded square on the grid are the closest to v_{max}. The three highlighted squares below the figure show the insertion possibilities that keep the route viable. We chose the configuration with the lowest cost, which in the case of the example, is the configuration shown in the central square. After the insertion, we update the analyzed segment and the neighborhood considered, remembering that we keep the la value stable. We repeat this process until adding more POIs to the route is no longer possible.

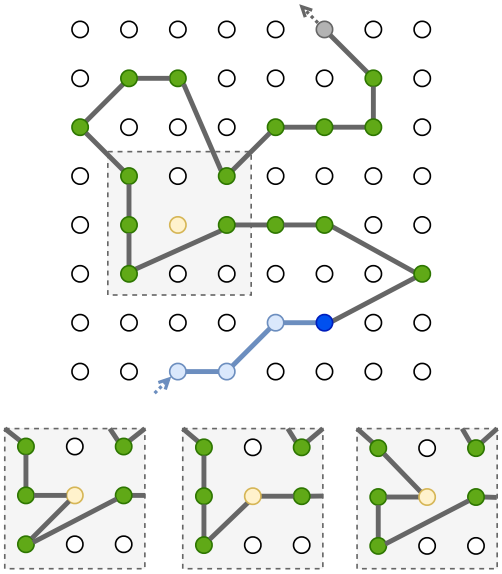


Figure 4.3: Example of possible configurations when inserting the yellow POI in the route. The squares below the figure show the POIs closest to the yellow one and the feasible configurations. The configuration chosen is the one with the lowest cost. In this example, the configuration of the middle square is the least expensive.

From the scenario presented in Figure 4.2f, we show an example of the POIs insertion step in Figure 4.4. In Figure 4.4a, the $la = 15$ green POIs represent the route segment that we will try to increase, while the region shaded in light green represents the neighborhood of the POIs defined by the parameter nr. In Equation 4.2, we only analyzed the POIs in the shaded region. The yellow node indicates the one found by solving Equation 4.2 and must be inserted in the route. In Figure 4.4b, we show the result of the POI insertion using the lowest cost configuration. We can see that the size of the analyzed segment remains 15 and that there was an update in the set of POIs that are part of the neighborhood. We carry out the insertion step until it is no longer possible to add a POI to the segment due to the lack of suitable POIs or the agent's budget. Figure 4.4c shows another iteration of the insertion step, where the POI indicated in yellow is the most profitable and should be inserted. If we go back to Figure 4.2 and consider that we are working on the same example, we notice that the yellow POI in Figure 4.4c was present on the route before the adaptation procedure. There is no issue reinserting this POI because if it was chosen by solving Equation 4.2, it means that it

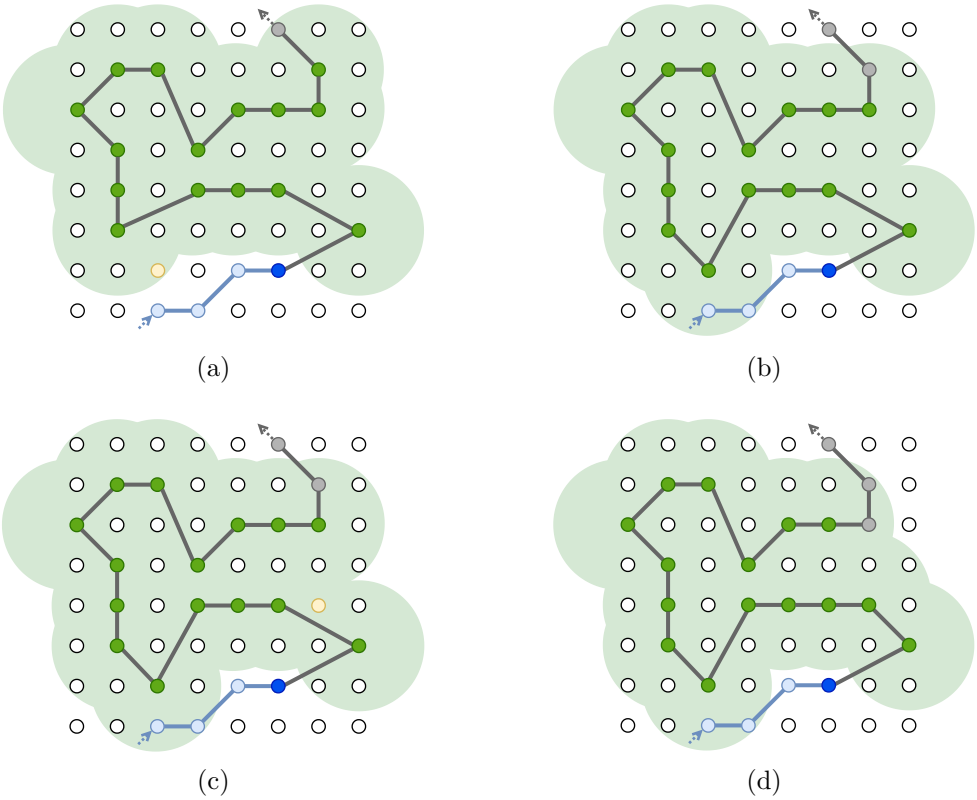
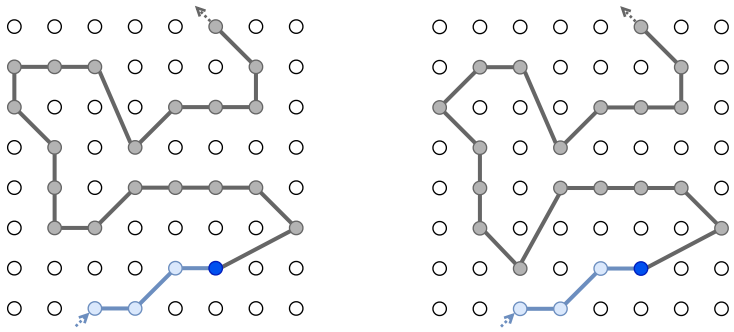


Figure 4.4: Illustration of the two insertions performed considering parameters $la = 15$ and $nr = 1.5$. The green POIs represent the segment of la POIs analyzed, the area shaded in light green represents the neighborhood delimited by nr, and the yellow POIs are those found by Equation 4.2 and that must be inserted.

is the one who most increases the agent's reward in any way. Figure 4.4d shows the final result of the two insertions and the update of the POIs into the segment and the neighborhood. It is worth mentioning that there is no relationship between the number of POIs removed and the number of POIs inserted in the segment. In the example we presented, three removals were performed, and then only two POIs were placed on the route without violating the constraints.

When adding a node to the route is no longer possible, we re-evaluate the cost-effectiveness of the agent's remaining route to compare it with the value before the changes. Figure 4.5 compares the route segment before and after the adaptation process. If the route before adaptation (Figure 4.5a) is more profitable, the adjusted route (Figure 4.5a) is discarded. Otherwise, the readjusted route overwrites the agent's original route. It is important to note that our algorithm performs all of its adaptation on a copy of the agent's route. This copy allows us not to stop the mission to make adjustments, ensuring that the route programmed for the agent to follow always feasible.

In this Section, we described the operation of Algorithm 1, equivalent to adjusting a route segment of one agent. When detecting that an agent does not fulfill the plan, our heuristic strategy consists of repeatedly applying Algorithm 1 whenever an agent changes



(a) (b)

Figure 4.5: Illustration of the route segment before (a) and after (b) the execution of Algorithm 1. The most profitable route will be assigned as the agent's current route.

position on its route. In the context of repeatedly applying the described procedure, we can say that our heuristic is an anytime algorithm. We detail this property in the next Section, where we also briefly analyze the complexity of our method.

4.2.1 Computational Cost Analysis

Since we use the heuristic proposed by Tsiogkas and Lane [66] in the planning phase, we will not analyze this method's behavior. In this Section, we present an analysis of the route adaptation method proposed in Section 4.2.

A feature that we must highlight in our heuristic is that it can be considered an anytime algorithm. An anytime algorithm gradually improves the quality of a solution according to its execution time and returns the current solution if it is interrupted [25]. In an anytime algorithm, it is possible to examine the solution at any time and choose to terminate the algorithm, temporarily suspend it, or allow the algorithm to run until the end. A route planning algorithm intended for practical use is usually limited by computational resources when it is to be used in real-time. Also, unforeseen events such as the failure of an agent further contribute to the problem's difficulty. In this scenario, the use of anytime algorithms is beneficial. They quickly provide viable planning, but not necessarily the ideal one, and can gradually improve the plan over time. A system based on anytime planning overlaps two functions over time: 1) the execution of the current plan or an initial part of it; and 2) the calculation of a new plan, or a pending part of it, to replace the current plan with an improved plan [36].

Regarding the time complexity of our method, we must take into consideration mainly the parameters: route size ($|\pi|$), size of the analyzed segment (la), and the size

of the neighborhood considered to choose a POI to be inserted in the route (nr). Let us look first at some essential functions for Algorithm 1:

- REWARD(v_i): computes the reward of v_i (Equation 3.1). This function does not appear explicitly but is called by the functions on the algorithm. The computation of the reward depends on the correlation range value. The correlation range defines the size of the set $\boldsymbol{\mathcal{N}}^{cr}_{v_i}$ of the neighbors of v_i considered when solving the equation. Thus, the reward calculation for v_i has $O(|\boldsymbol{\mathcal{N}}^{cr}_{v_i}|)$ complexity.
  - worst-case: an embracing correlation range that covers all the instance POIs, that is, $|\boldsymbol{\mathcal{V}}|$. Thus, the worst-case complexity is $O(|\boldsymbol{\mathcal{V}}|)$.
- EVALUATE(π, pos): computes Equation 3.1 for the POIs from the agent's current position until the end of the route. That is, apply the REWARD(v^{π}_i) function considering all $pos < i < |\pi|$. Therefore, the EVALUATE( ) function complexity is $O((|\pi|\text{-}pos\ )\times|\boldsymbol{\mathcal{N}}^{cr}_{max}|)$, where $\boldsymbol{\mathcal{N}}^{cr}_{max} = \max(\{|\boldsymbol{\mathcal{N}}^{cr}_{v^{\pi}_i}| \mid pos < i < |\pi|\})$.
  - worst-case: an agent fails at the starting point of the route. Thus, we will have to execute Algorithm 1 at the beginning of the route of the agents who remain active in the mission. We have $pos = 0$, soon $|\pi| - pos = |\pi|$. When we also consider the worst-case of the REWARD( ) function, we have that the worst-case complexity of the EVALUATE( ) function is $O(|\pi|\times|\boldsymbol{\mathcal{V}}|)$.
- REMOVELEASTCONTRIBUTOR(π, pos, la): removes the least profitable POI in the segment. First, we have to find the POI and then remove it from the route. The removal operation has a constant cost. To find the POI to be removed, we have the cost of $O(la\ \times|\boldsymbol{\mathcal{N}}^{cr}_{max}|)$, where $\boldsymbol{\mathcal{N}}^{cr}_{max} = \max(\{\boldsymbol{\mathcal{N}}^{cr}_{v^{\pi}_i} \mid pos < i \leq pos + la\})$, since we need to evaluate each one of the la POIs using the REWARD( ) function. However, in Algorithm 1, this function is performed $rem\%$ times. Therefore, in Algorithm 1, the function has complexity $O((\frac{rem}{100}\times la\ )\times la\ \times|\boldsymbol{\mathcal{N}}^{cr}_{max}|)$.
  - worst-case: the agent is at the beginning of the route and $la = |\pi|$, and we remove the entire segment, that is, $rem = 100$. Replacing la with $|\pi|$ and making $rem = 100$, we have $O((\frac{100}{100}\times|\pi|)\times|\pi|\times|\boldsymbol{\mathcal{N}}^{cr}_{max}|) = O(|\pi|\times|\pi|\times|\boldsymbol{\mathcal{N}}^{cr}_{max}|)$ $= O(|\pi|^2\times|\boldsymbol{\mathcal{N}}^{cr}_{max}|)$. Finally, considering the worst-case of the REWARD( ) function, we have that the worst-case complexity of the REMOVELEASTCONTRIBUTOR( ) function, in Algorithm 1, is $O(|\pi|^2\times|\boldsymbol{\mathcal{V}}|)$.
- INSERTFITTEST($\pi, pos, la, nr, \beta_k$): inserts the most profitable POI in the segment. For each v_i of the segment, evaluates if the neighbors are viable and the reward they offer. To check the viability of a POI has a constant time. To calculate the reward of the POIs in the segment we have $O(la\times|\boldsymbol{\mathcal{N}}^{nr}_{max}|)$, where $\boldsymbol{\mathcal{N}}^{nr}_{max} =$

$\max(\{\ |\mathcal{N}^{nr}_{v^{\pi}_i}|\ \mid\ pos\ <\ i\ \leq\ pos\ +\ la\})$. We consider the closest POIs to insert the selected one into the route. To find the nearest POIs, we spend time $O(|\mathcal{V}|)$. Therefore, the InsertFittest( ) function has time complexity $O(la{\times}|\mathcal{N}^{nr}_{max}|{+}|\mathcal{V}|)$ $= O(la{\times}|\mathcal{N}^{nr}_{max}|)$. Unlike the RemoveLeastContributor( ) function, we do not know in advance how many times the InsertFittest( ) function will be executed. Assuming that we are analyzing an optimal segment, the method must insert the same POIs removed, as there are no POIs better than them near the segment. We can say then that in the Algorithm 1 the InsertFittest( ) function has time complexity $O((\frac{rem}{100}{\times}la){\times}la{\times}|\mathcal{N}^{nr}_{max}|)$.

- worst-case: the agent is at the beginning of the route and $la = |\pi|$, and we removed the entire segment at the last step, that is, $rem = 100$. Replacing la with $|\pi|$ and making $rem = 100$, we have $O((\frac{100}{100}{\times}|\pi|){\times}|\pi|{\times}|\mathcal{N}^{nr}_{max}|) = O(|\pi|{\times}|\pi|{\times}|\mathcal{N}^{nr}_{max}|) = O(|\pi|^2{\times}|\mathcal{N}^{nr}_{max}|)$. Finally, considering the worst-case of the Reward( ) function, we have that the worst-case complexity of the InsertFittest( ) function, in Algorithm 1, is $O(|\pi|^2{\times}|\mathcal{V}|)$.

Table 4.1 summarizes the complexities of the main functions of Algorithm 1. Taking into account that the Reward( ) function is called by the other three functions analyzed and its complexity is already reflected in the complexity of the other three, one complete execution of Algorithm 1 has, in the worst-case, time complexity $O(|\pi|{\times}|\mathcal{V}|) + O(|\pi|^2{\times}|\mathcal{V}|) + O(|\pi|^2{\times}|\mathcal{V}|) = \max\{\ O(|\pi|{\times}|\mathcal{V}|), O(|\pi|^2{\times}|\mathcal{V}|), O(|\pi|^2{\times}|\mathcal{V}|)\} = O(|\pi|^2{\times}|\mathcal{V}|)$.

Table 4.1: Summary of the main functions complexity of Algorithm 1 depending on the elements involved and, in the worst-case scenario.

Function	**Complexity**	**Worst-case**
Reward	$O(\lvert\mathcal{N}^{cr}_{v_i}\rvert)$	$O(\lvert\mathcal{V}\rvert)$
Evaluate	$O((\lvert\pi\rvert\text{-}pos\ ){\times}\lvert\mathcal{N}^{cr}_{max}\rvert)$	$O(\lvert\pi\rvert{\times}\lvert\mathcal{V}\rvert)$
RemoveLeastContributor	$O((\frac{rem}{100}{\times}la){\times}la{\times}\lvert\mathcal{N}^{cr}_{max}\rvert)$	$O(\lvert\pi\rvert^2{\times}\lvert\mathcal{V}\rvert)$
InsertFittest	$O((\frac{rem}{100}{\times}la){\times}la{\times}\lvert\mathcal{N}^{nr}_{max}\rvert)$	$O(\lvert\pi\rvert^2{\times}\lvert\mathcal{V}\rvert)$

Since our method consists of applying Algorithm 1 repeatedly as the agent reaches a new POI on its route. In the worst-case scenario, the adaptation routine starts to be executed at the beginning of the agents' route and is therefore called $|\pi|$ times. Thus, we can say that the complexity of using our method on a monitoring mission is $|\pi|{\times}O(|\pi|^2{\times}|\mathcal{V}|) = O(|\pi|^3{\times}|\mathcal{V}|)$.

As much as the complexity seems high, the worst-case rarely happens because we limit the correlation range, the set of candidate POIs to be inserted in the route, and we generally consider a value for la smaller than the size of the route.

4.2.2 Robotic system deployment scheme

This Section shows a baseline deployment scheme for a system that performs the proposed adaptation routine in the mission execution phase. The most direct implementation uses a central processing point responsible for coordinating the team and synthesizing the collected rewards. In Figure 4.6, we see this central processing point represented by the antenna at the top left communicating with the team formed by three robots. In summary, the central point periodically sends the updated route for each agent. The robots, in turn, run the route and send the information collected at the points of interest and the status of the mission to the central base. The lower part of Figure 4.6 shows more details about the behavior of each of the agents in the system.

The central point is the brain of the monitoring system, ensuring that the collected data is transformed into valuable information that provides insights for users. For this, the control point should be kept open a bidirectional, reliable, and high-speed communication channel to exchange information in real time with the robots during the entire execution of the mission. Without loss of generality, we disregard possible interferences and other technical aspects. The only consideration at this point is the case where the connection between the robot and the central base is lost, as indicated by the red arrow in Figure 4.6. In these cases, we consider the robot to have failed, and the central base starts to adapt routes of the still-active robots.

The robots, in turn, must also maintain effective communication with the processing center. In addition to following the routes received periodically, robots must be able to collect information at a point of interest and transmit it to the central point with attached location tags. Additionally, it is desirable that the robots be equipped with mechanisms to detect failures or issues affecting their operational capacity. If a robot detects a failure or realizes it cannot complete its route, this information must also be transmitted to the processing base that will make a decision.

The rewards are all computed and summed in the central processing point based on the data received by the robots. This strategy keeps the monitoring status current and prevents information collected by failed robots from being lost. At any time, a robot can stop responding or report that a failure has occurred. As the system knows the routes and speeds of the robots, it is possible to estimate a timeout that indicates that the robot has failed and is no longer responding. At that point, the central base activates the adaptation routine, deactivates communication with this robot as if it were being withdrawn from the mission, and starts to send modified routes to the still active robots.

These requirements are a starting point for the area monitoring system to operate with the proposed adaptation routine. The system can become more robust by following these steps and continuously refining it based on real-world usage and feedback.

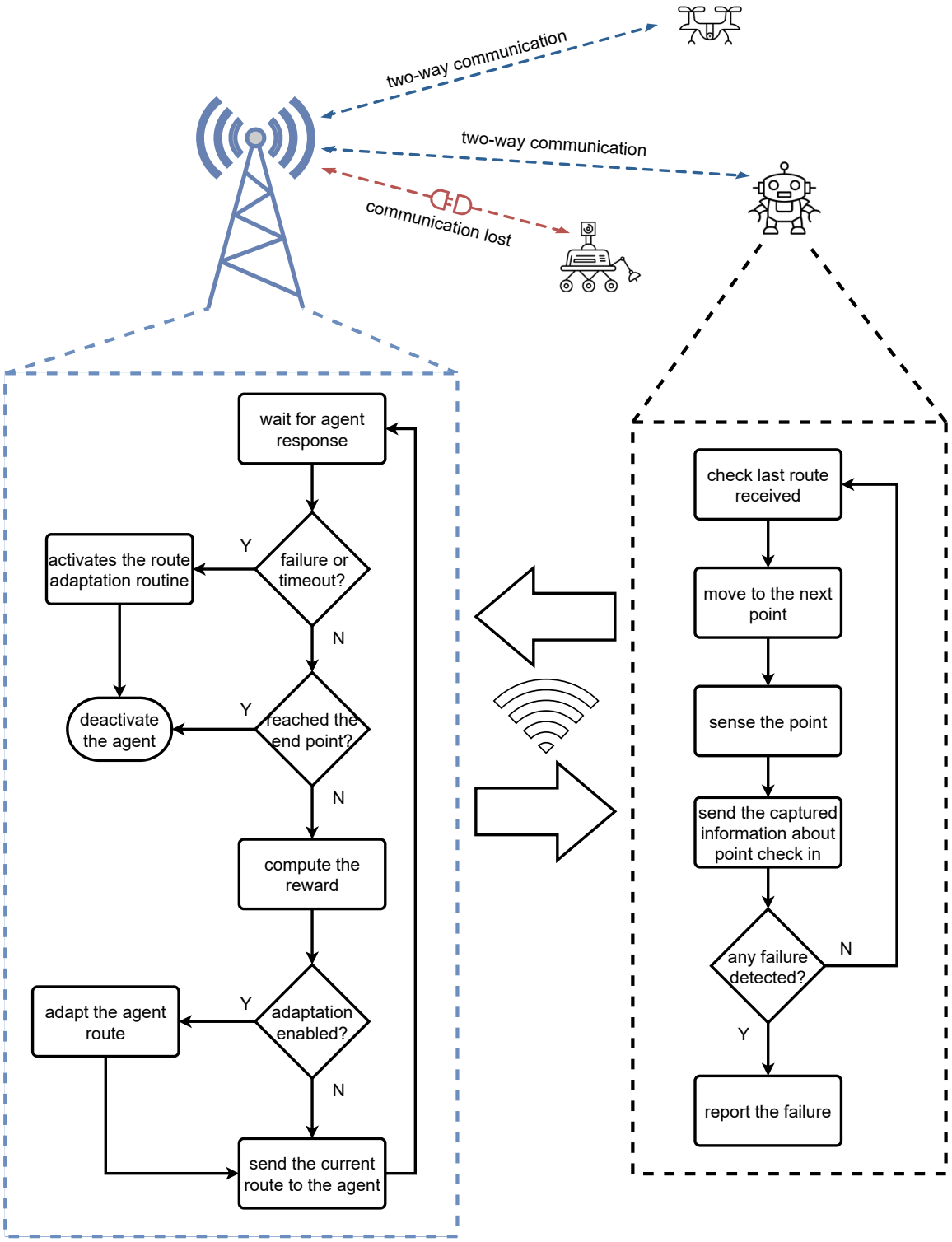


Figure 4.6: Baseline deployment scheme of our adaptation routine in a real monitoring mission. The top part shows a central base maintaining two-way communication with all robots. The lower part exemplifies the behavior of each system agent during the mission.

4.2.3 Experiments

This Section presents the experiments to evaluate the proposed method's effectiveness for adapting routes. We assessed the impact of applying our method in sensing mission simulations. A simulation consists of following an initial plan obtained for a set of POIs and a team of agents, where one or more agents do not fulfill the plan assigned to it for any reason. These experiments helped us understand the problem better and outline the objectives achieved with this project.

Conduction of the experiments

We provide a method of adapting routes applied during the mission's execution. As we do not know an optimal solution for the instances proposed in Section 3.1.1, we need a way to find initial planning that is at least a sub-optimal solution. In these experiments, we computed the initial plan using the genetic algorithm proposed by Tsiogkas and Lane [66] using the default parameters. We chose this heuristic because it was designed specifically for CTOP, and the authors indicated its potential to be applied in online situations.

Without loss of generality, we consider that the distances between POIs are given in meters, and the costs related to the execution of any task are given in seconds. For all agents, the cost of moving from one point to another is proportional to the Euclidean distance between them, i.e., we consider the agents' velocity as 1 m/s. Another cost is capturing the reward of a POI, which we assume the value of 1 second for all agents. Besides, we consider that all agents have the same time budget to accomplish the mission. To set the budget for each agent, we use the same strategy used by Tsiogkas and Lane [66]. First, we defined the maximum budget as the lowest cost for a single agent to go through all the POIs and collect all the rewards. Then, we divided the maximum budget by the number of agents in the team and set the budget for each agent as 75% of this value.

Finally, we assume all agents capture the same amount of correlated reward from neighboring POIs. The correlation $\phi(\cdot)$ between POIs used to extract the weights w_{ij}, used in Equation 3.1, is given by the exponential decay function $\phi(v_i, v_j) = e^{-\lambda\|v_i - v_j\|}$, where λ determines how fast the correlation of two points reduces with the distance. Generally, the distance where two points can be correlated is more significant when λ is bigger. During the experiments, we adopted $\lambda = 1$.

As the GA can generate different solutions each time it runs, we run the algorithm ten times for each instance and choose the solution with the highest planned reward for the team as the initial plan. Based on this plan, we place all agents at the starting point, and the mission simulation begins. At each iteration, one of the agents moves to the next position indicated on its route. One agent at a time, always following the same order of

action until everyone reached the endpoint of the mission. However, we are interested in scenarios where we are unsure whether the agent's behavior will be as planned. To reproduce the worst-case of this situation, when the agent fails and can no longer sense the environment, we exclude an agent from the mission at a certain point in the simulation. At that point, part of the POIs will no longer be visited, impairing the team's final reward collection. As a way of comparing whether our strategy is efficient, in all tests, we consider the following strategies:

- **Non-adapt:** ignores the failures, with each remaining agent following its initially planned routes;
- **Replanning:** given the current state of the mission, i.e., agents and locations left to be visited, rerun the meta-heuristic proposed by Tsiogkas and Lane [66];
- **Ours (adapt):** applies the anytime method that we proposed to adapt the current routes.

The test environment consisted of a machine equipped with an Intel XEON E5-2650 CPU, 256GB of RAM, and Ubuntu 16.04. The GA proposed by Tsiogkas and Lane was implemented in C++[1], and our adaptation routine was coded in Python 3.5.

Illustrative example

Figure 4.7 shows an example of an instance with 225 POIs organized in a grid and one region more important to inspect in the middle of the grid. This region should be prioritized in possible failures of the agent responsible for sensing it. Figure 4.7b depicts the initial routes planned using the GA for a team with three agents. In this example, the red agent was accountable for covering the entire region with more valuable POIs.

In an ideal scenario, the three agents would strictly follow the routes assigned to each. If this occurs, the final reward collected by the team would be $2,124.899$. However, if the red agent fails before the end of the mission, the final reward collection will be drastically affected. As this agent is solely responsible for inspecting the central region of the environment, which is the most valuable, the team's success in the mission depends a lot on how the red agent will behave. If it fails before reaching this region, and the other two agents ignore this failure, the team would collect a final reward equal to $1,426.167$ at the end of the mission, about 67% of what was planned.

To recover part of the reward that would be lost due to the red agent's failure, Figure 4.7c shows the result of reapplying the GA and Figure 4.7d shows the effect of using our adaptation method. In reapplying the algorithm used in the initial plan generation, the simulation must be stopped until the algorithm calculates the new routes. With this

[1]Publicly available by the authors at https://github.com/lounick/lwga

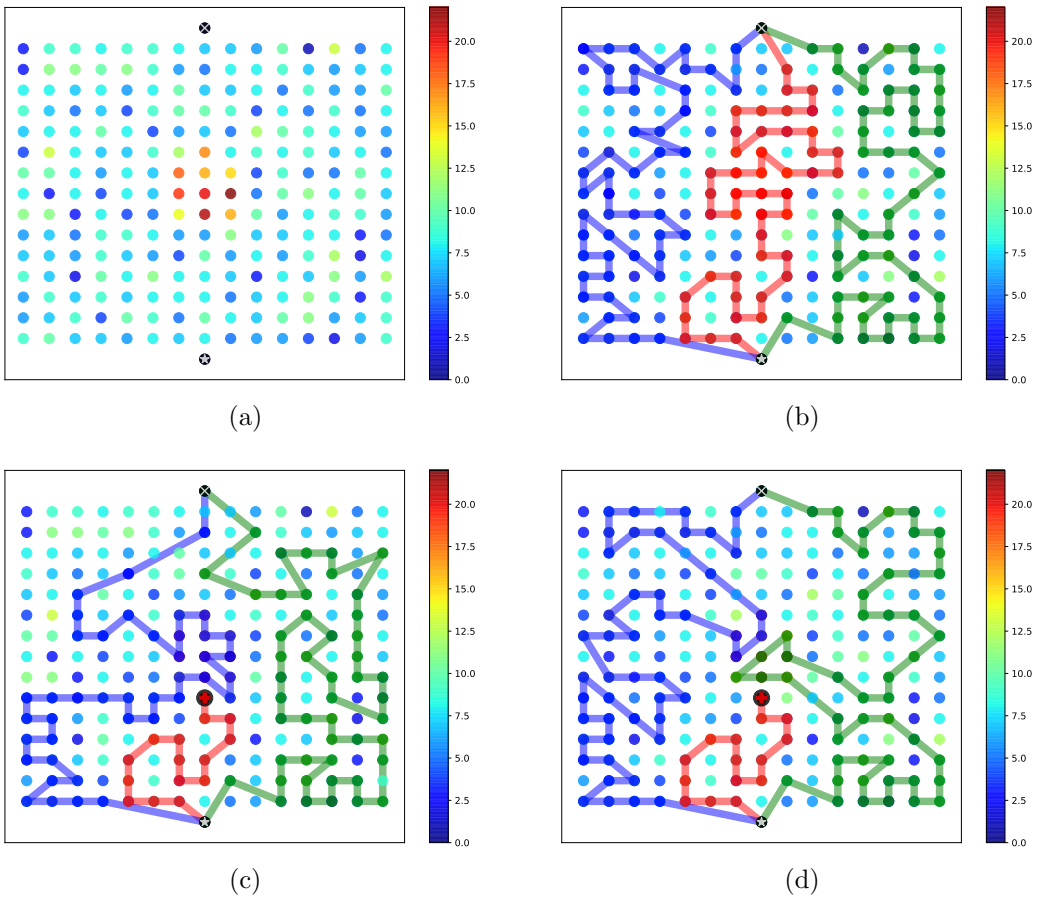


Figure 4.7: An illustrative example of an instance with 225 POIs with one region more important to inspect at the center. (a) Grid POIs layout. The redder the POI, the more reward it offers. The agents leave from the bottom and must go to the top. (b) The initial set of routes considering three agents. (c) The red agent fails during the mission, and the routes are replanned using Tsiogkas and Lane method. (d) Result by applying our route adaptation method.

strategy, the plan executed in Figure 4.7c generates a final reward of $1,703.686$, about 80% of the initially planned. Using our method, since we consider the current routes, the simulation can continue to evolve, and the agents will always have a valid route to follow. With our adaptation routine, the final reward collected by the team would be $1,822.533$, equivalent to almost 86% of what was planned for the mission.

In addition to recovering almost 6% more final reward than GA, we can see that our approach does not completely dismantle the agent's initial route, as with the blue agent. In Figure 4.7c, it practically abdicates the regions assigned to it at the beginning of the mission. While in Figure 4.7d, with our method, the blue agent does not entirely give up what was assigned to it and receives the help of the green agent to cover the region previously assigned to the red agent. Also, taking advantage of an already planned route made our method spend less than half the time than replanning the agents' routes who continued on the mission.

Initial plan optimization

The strategy used to generate the initial mission planning is a meta-heuristic and might produce sub-optimal solutions. Aiming to minimize the impact of these sub-optimal solutions in our analyzes, we further improve the initial planning by also applying our algorithm at the beginning. To optimize the initial plan, we simulated the mission's execution for the planning, as described in Section 4.2.3, but without removing any agent during the execution of the mission. Besides, our algorithm starts to adapt the routes from the starting point of the routes. As our method depends on the routes' configuration to make the adjustments, it is sequentially executed five times to ensure sustainable results.

Table 4.2 shows that the proposed method is beneficial even when there are no failures, increasing the initial planning calculated by the GA. Additionally, it is worth noting the significant decrease in the processing time reported in the last two columns of the table. The Plan column is the average time spent generating the instance's initial solution using the GA. In contrast, the Opt column shows the sum of the five executions performed to optimize the plan with our method.

Table 4.2: Comparison between the initial plan obtained by the GA and the one optimized by our method for different instance sizes. These results consider a team with five agents.

Amount of POIs	**Total reward**		**Gain**	**Avg Time (sec.)**	
	Plan*	**Opt.**	**(%)**	**Plan***	**Opt.**
150	1,381.250	1,383.529	0.2	6.13	1.05
225	3,479.750	3,527.320	1.4	15.12	8.47
400	5,119.470	5,261.795	2.8	41.69	21.29
625	7,915.880	8,182.674	3.4	101.52	68.01
900	9,805.710	10,023.417	2.2	227.73	126.93

* Using the GA provided by Tsiogkas and Lane [66].

We see in Table 4.2 that as we increase the number of POIs, the greater the gain of using our method to optimize the solution computed by GA. We believe this phenomenon occurs because we use the GA with the default parameters, which must have been calibrated for the experiments by Tsiogkas and Lane, considering instances with a maximum of 81 POIs [66]. Despite this, our method proved effective in optimizing an initial solution calculated for the CTOP without spending too much time on this task.

Results and discussion

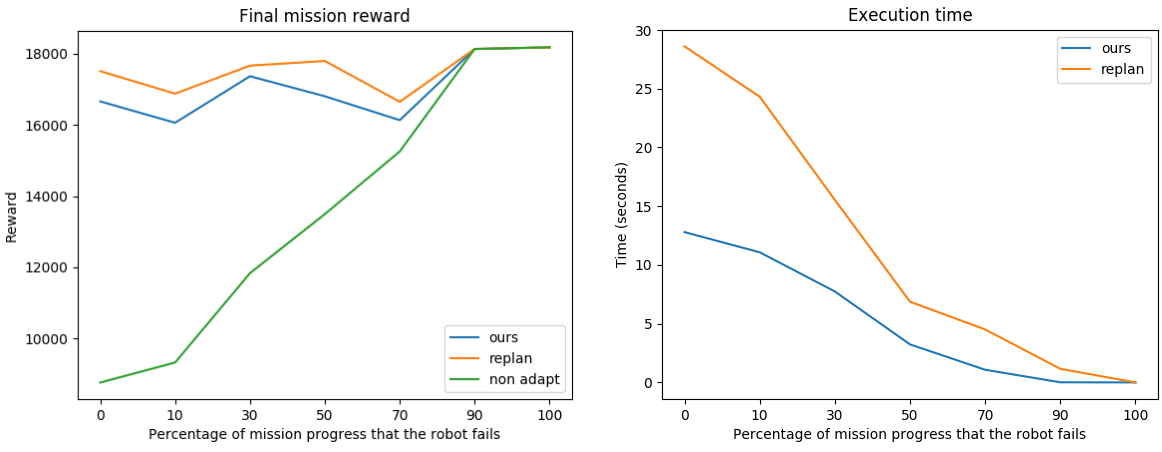
Given the optimized initial plan, we seek to analyze the reward our method would get to recover when an agent does not fulfill what was assigned to it in the mission planning. We performed several trials varying the instance size, the number of agents in the team, and

the route percentage by which the selected agent leaves the mission. We also evaluated the impact of different route segment lengths and the neighborhood range for the adaptation procedure. We use the three strategies described in Section 4.2.3 in each trial.

The replanning strategy demands the mission to stop as an entirely new plan must be determined. We generate a new instance considering only the POIs not yet visited. To guarantee the routes continuity and the agents' budget limit, we make all the POIs unreachable from the starting point, except the agents' current POIs. Then, we set the cost to reach these POIs as each agent's spent budget amount until the mission's current state. After generating a solution for the new instance, we merge the paths already taken in the mission with the new planned routes so that the agents can continue the mission considering these new routes.

Unlike the replanning approach, our algorithm adapts the route section closest to the agent without stopping the mission. Once a failure occurs, the algorithm begins to be applied to all routes of the remaining agents in the mission. Each time the agent moves from one POI to another, we execute the algorithm to update the next route segment until the agent finishes the mission. Notably, from the moment the algorithm starts to be executed, the anytime property guarantees that it always maintains a viable solution based on the initial plan.

Figure 4.8 compares the three strategies on an instance with 225 POIs and a team with three agents. To verify how the algorithm responds at different mission stages, we forced one agent to fail at the beginning of the mission or after completing a percentage of its route. To create the worst-case scenario, we remove the agent with the route with the highest reward in the initial planning.



(a) (b)

Figure 4.8: Comparison for a team of three agents in an instance with 225 POIs when the agent with the highest planned reward fails in a certain percentage of the plan.

The impact of removing a single agent at the beginning of the mission and not taking any action can be significant. Both strategies that modify the initial plan can recover

a large portion of the reward that would be lost, with comparable results (Figure 4.8a). As expected, completely recalculating a new plan achieves a higher reward, but not significantly higher than our approach. However, our method performs considerably faster (Figure 4.8b), which makes it much more applicable. It is worth noting that the time shown in Figure 4.8b is the additional time spent executing the methods. Besides, the time spent by our strategy is diluted during the mission execution, unlike the replanning strategy, where the indicated time is related to the time the simulation waits for the new routes calculation.

Table 4.3 shows the comparison of variation of the adapted segment' size (la) and the neighborhood range (nr) for an instance with 900 POIs and a team with three agents. The agent with the highest planned reward fails at the beginning of the mission. The last two lines of the table show the values obtained when ignoring the failure and choosing to replan the two remaining agents' routes. Theoretically, the higher the values of la and nr, the better the chances of acquiring a good adaptation. That way, the heuristic considers all not visited POIs and adapts in any position on the rest of the route. However, the values in Table 4.3 show that, in practice, this is not always the best option.

Table 4.3: Comparison of different values of la and nr for a team with three agents in an instance with 900 POIs when the agent with the highest reward in the initial plan fails at the beginning of the mission.

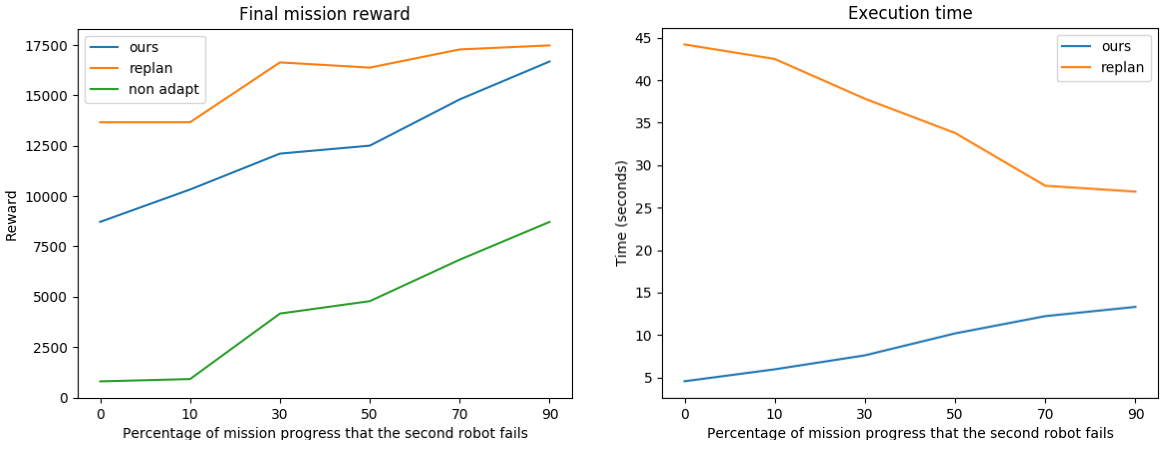
la	nr	Reward (%)	Time (sec)	POIs	Mean time
	2	**82.89**	**60.14**	**169.50**	**0.18**
10	4	75.62	68.93	168.50	0.21
	8	81.05	76.91	160.00	0.24
	2	**91.47**	**157.90**	**169.00**	**0.47**
20	4	90.98	133.20	167.50	0.40
	8	90.85	219.12	165.50	0.67
	2	**90.85**	**309.18**	**161.50**	**0.97**
30	4	90.71	278.11	162.50	0.87
	8	90.63	370.77	160.00	1.17
until	**2**	**90.04**	**5,934.60**	**150.50**	**19.98**
the	4	88.60	3,917.76	131.00	15.19
end	8	88.48	4,234.11	130.00	16.54
Non-adapt		52.59	0.11	173.50	–
Replanning		91.14	401.86	185.00	–

Our heuristic is designed to apply small adjustments in the route to cover regions abandoned by other agents without forgetting the initial planning. We risk deconfiguring the initial route and performing poorly in reward recovery by using high values for la and

nr parameters. It is then expected that the best values for these parameters keep the route close to the region where it was initially planned. Following this intuition, the best results in the table are those using $nr = 2$, precisely the value considered in the utility function evaluation to calculate the correlated reward in the planning method.

The POIs column shows the average number of POIs visited by the two agents that completed the mission. Our method tends to decrease the number of POIs in the routes. However, having routes with fewer POIs does not imply an under-utilization of the budget. On the contrary, this behavior frees up the budget for the agent to visit important POIs uncovered by failed agents. The last column in Table 4.3 lists the average execution time for a call to our method. It is worth mentioning that our method's execution time does not depend on the number of POIs in the problem as a whole but only on the number of POIs in the path and the size of the analyzed segment.

We also analyzed the case with more than one agent failing during the mission. Figure 4.9 compares the three strategies on an instance with 225 POIs and a team with three agents. Again, the agent with the highest reward in the initial plan fails at the beginning of the mission to enforce the worst-case scenario. Figure 4.9a shows the reward captured by the team at the end of each mission when the second agent fails at different plan progress percentages. We can notice that stopping the mission and replanning is the best option for recovering the reward that would be lost. However, in this situation, the mission has to be interrupted twice, leading to unnecessary expenditure of energy and time while the agents wait to calculate the new routes.



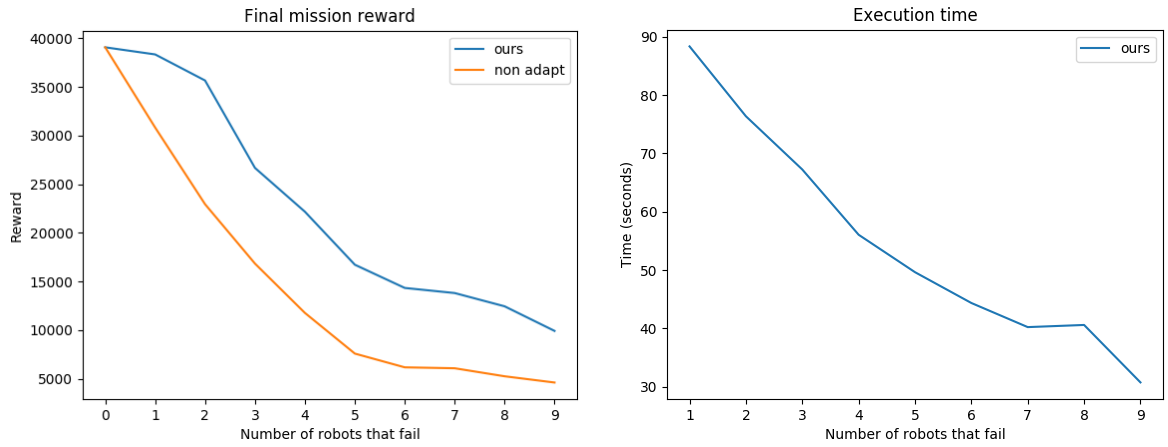
(a) (b)

Figure 4.9: Comparison for a team with three agents in an instance with 225 POIs when the two agents with the highest planned reward fail. The profitable agent fails in the mission starting while the other fails in a certain percentage of the plan.

Figure 4.9b shows that the longer the second agent takes to fail, the mission execution time with our algorithm increases. While using the routes replanning strategy, the mission completion time decreases. This variation occurs because the two algorithms'

execution time depends on the number of POIs evaluated in the problem. The closer the agents are to completing the mission, the smaller the number of non-visited POIs, and the replanning algorithm calculates a solution more quickly than at the mission's start. Regarding our method, since we apply it repeatedly to all active agents' routes in the mission, the longer we stay with agents on the mission, the more times we execute our algorithm, thus increasing the mission's total time. Nevertheless, the time spent by our method is almost half the time spent replanning the routes.

To test our method in a scenario where replanning would be impracticable, we simulated one mission with 900 POIs and a team with 10 agents. Figure 4.10 shows that using our method makes it possible to recover part of the reward the team would lose with little processing time. In this experiment, the agent with the highest planned reward fails at the beginning of the mission, and for every 5% plan progress, another agent is forced to die. For instance, when nine robots fail, the ninth agent will fail on 45% of the plan's progress, and the remaining agent will be alone for more than half of the mission. Results shown in Figure 4.10b reinforce that our method depends not on the instance size but on the number of agents acting in the mission. We can see that the more agents fail, the shorter the simulation time using our route adjustment algorithm.



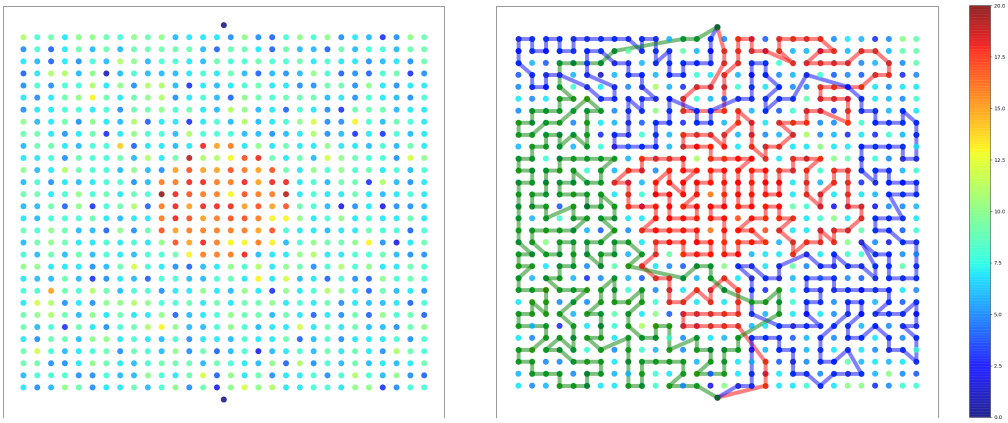
(a) (b)

Figure 4.10: Comparison for a team with ten agents in an instance with 900 POIs when up to nine agents can fail. The first agent fails at the beginning of the mission, and for every 5% of plan progress, another agent fails.

Together these results demonstrate that our method is a robust and appropriate option for cases where agent failures are possible. Another advantage of our method is its anytime property. Thus, we can activate it depending on the situation or even let it run for time intervals. While the algorithm runs, the agents always maintain a viable route to follow and have pieces of their routes adjusted to optimize the collected reward.

Significant insights

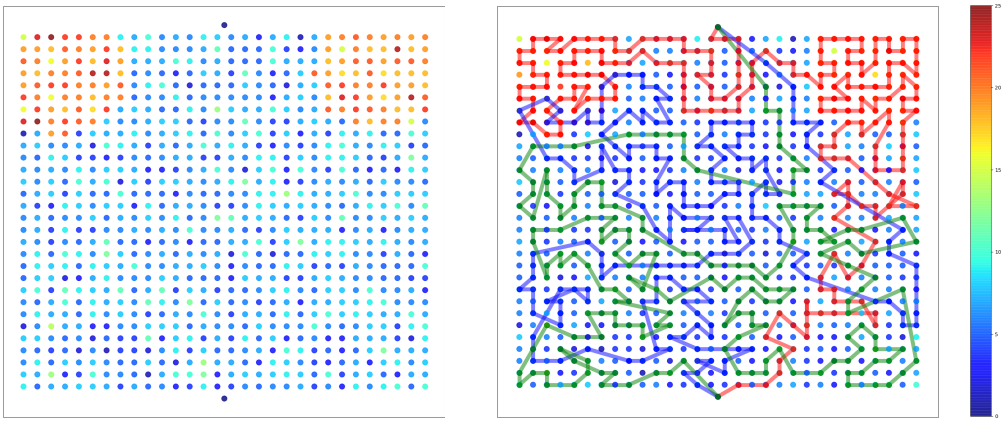
We realized that most of the works that address TOP and its variants are concerned only about adding the maximum possible reward in the mission without considering the load assigned to each of the team's agents. In this way, we can have situations where an agent is responsible for collecting the highest percentage of rewards. We can see in Figure 4.11 that the agent that follows the red route is responsible, alone, for sensing the central region of the instance, which is the most valuable region in the environment. Should this agent fail or abandon the mission, the team's final reward must fall sharply.



(a) (b)

Figure 4.11: Example of unbalanced routes: (a) instance with one region more important to inspect at the center; (b) route planning for three agents generated using the GA.

In Figure 4.12, the problem that this imbalance can cause is even more significant. The instance has two regions with more valuable POIs that deserve further attention and are relatively far away. However, the traced routes show that the same agent will be responsible for visiting these two regions, further increasing the team's loss if it fails.



(a) (b)

Figure 4.12: Example of unbalanced routes: (a) instance with two regions more important to inspect; (b) route planning for three agents generated using the GA.

We showed in Section 4.2.3 that our method can recover part of the reward that would be lost if these agents do not comply with the plan without the need to spend much time. However, when we predict a situation like this, we desire a close sum of rewards collected by each agent or that the regions with greater importance are not under only one agent remit. Because in case of failure of any agent, the impact on the team's final reward will be smaller, and the teammates will be able to adapt their routes in the best way to cover these areas. In this sense, we propose in the next section a planning methodology that seeks to generate initial plans prepared for agent failure cases.

4.3 Planning for failure-prone teams

Once we confirm that adjusting the routes of the active agents in the mission allows us to cover part of the area that would be neglected by the failure of one or more agents, we propose a route planning method that benefits this strategy while solving the issues of unbalanced routes identified in Section 4.2.3. We are interested in environments with regions with different importance to inspect, so we divided our route planning strategy into two steps: regions and POIs. Algorithm 2 shows an overview of the proposed method, remembering that $\boldsymbol{\mathcal{V}}$ is the set of POIs, and $\boldsymbol{\mathcal{A}}$ is the set of agents.

Algorithm 2: MissionPlanning($\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{A}}$)

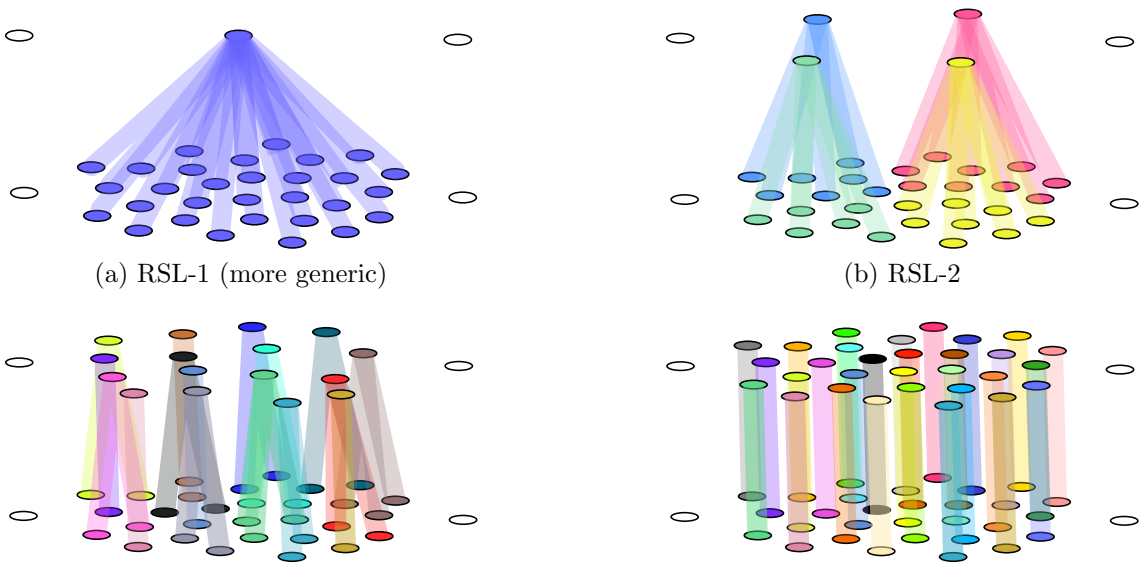
```
1 RSL ← CreateRegionSpecificityLevels(V)
2 RSL ← EstimateRSLValues(RSL, V, A)
3 Φ'← RegionLevelPlanning(RSL, A)
4 Φ ← POILevelPlanning(Φ', A)
5 return Φ
```

The CreateRegionSpecificityLevels function subdivides the POIs into several regions and creates a representation in specificity levels, while the EstimateRSLValues function estimates the values of rewards and costs for the created regions and levels. Section 4.3.1 describes how these two functions create regions and estimate values for the planning phase. The RegionLevelPlanning function is responsible for the first planning phase, where it must direct agents to the regions with the most valuable POIs in the environment. This step allows more than one agent to pass through the same region, as described in Section 4.3.2. Intuitively, the most important regions will have more than one agent passing through it so that one agent can cover it in case of teammate failure. In the second planning phase, we calculated a solution for CTOP, limiting each agent's accessible POIs according to the regions for which it was responsible in the previous

step. This step is done by the POILevelPlanning function, described in Section 4.3.3. Finally, Section 4.3.5 describes the experiments to validate the method.

4.3.1 Region Specificity Levels (RSL)

Before planning the routes, we create what we call Region Specificity Levels (RSL). Once we discretize the environment into a finite set of POIs, we assume that a region is a subset of points in that environment. Thus, we define several levels of environment representation, where the first is the most generalized, and the last is the most specific. Figure 4.13 illustrates an example of RSL with four levels, representing an environment with 30 POIs. In a level, each region is represented by the centroid of the points within it. The upper part of Figures 4.13a to 4.13d shows the region's representation created by the function CreateRegionSpecificityLevels. In contrast, at the bottom of these figures, we can see the projection of these centroids onto the points they represent in the original environment. The white nodes in the figures represent the start and end points, which remain unchanged at all levels. Figures 4.13b and 4.13c show two intermediate levels, while Figure 4.13d illustrates the most specific one. In this level, each region comprises a single POI, representing exactly the original environment.



(a) RSL-1 (more generic) (b) RSL-2

(c) RSL-3 (d) RSL-4 (more specific)

Figure 4.13: Region Specificity Levels for an environment with 30 POIs. The region representations are at the top of each sub-figure. For illustration purposes only, we projected the points representing a region onto the POIs of the original environment that belonged to it. The colors were chosen without specific criteria, only to highlight different areas.

The first level is the one that has only one region with all the POIs of the environment. From this level, we specify the regions by subdividing each existing region into four others. To do this, we ignore the starting and ending points of the routes and draw the smallest rectangle that limits all other POIs in the region. Once done, we project a Cartesian plane's origin onto this rectangle's central point. This way, each quadrant in this projected Cartesian plane is treated as a new region. We repeat this process until all regions have only one POI, ignoring regions that become empty. Figure 4.14 illustrates this process.

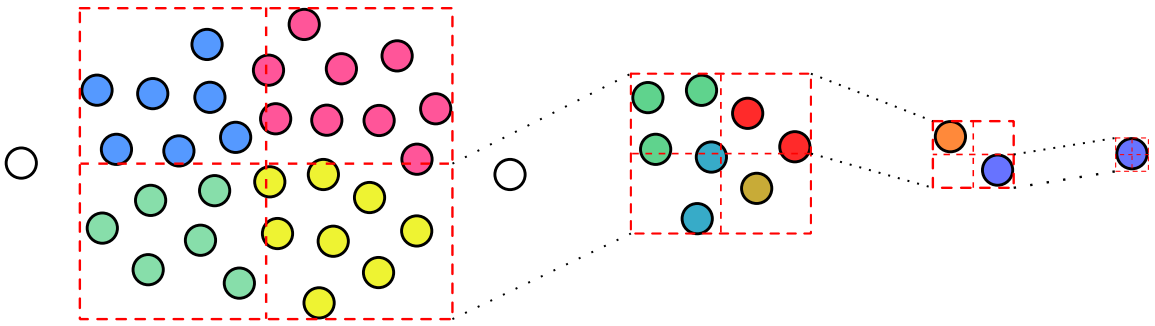
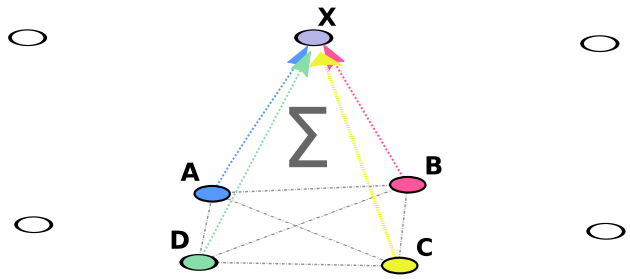


Figure 4.14: Process of dividing the environment into regions. The red dashed lines indicate the Cartesian plane projected onto the points dividing them into four regions. The same process applies to each new region until a region is empty or has only one POI. Colors follow the example illustrated in Figure 4.13.

In creating the RSL, we only consider the location of each POI. However, to plan, we need to know the cost of visiting a region and how much reward each area can offer. The EstimateRSLValues function is responsible for estimating these values. While creating the levels, we start from the most generic region to the more specific ones; we follow the opposite direction to estimate the costs and rewards of each region at each level. We adopt this strategy because the most detailed level is equivalent to the original environment without the proposed region generalizations. Thus, the cost and reward for each region at this level are the same as the own POI.

As we know the reward and cost of sensing each POI at the last level, we can combine these values to estimate those at a higher level, which is nothing more than a merge of more specific regions. In this way, assuming that the most generic level is level 1, to compute the rewards of a region on level x, we sum the rewards of each point on level $x + 1$ that make up this region on level x. To estimate the costs, we also sum the sensing costs to the points of the sub-regions. However, we must consider that visiting a region has an additional cost, given the displacement between the POIs that compose it. To compute this move cost, we generate all two by two combinations of the points on the more specific level. Then we calculated the average cost of moving between each pair of sub-regions and multiplied it by the number of combinations generated. We can see in Figure 4.15 an illustration of this process.



(a)

POI	Estimated		Distance to			
	Reward	Cost	A	B	C	D
A	121.3	253.7	0.0	16.0	21.1	10
B	133.4	225.2	16.0	0.0	12.4	23.6
C	77.0	197.3	21.1	12.4	0.0	15.2
D	98.6	210.4	10.7	23.6	15.2	0.0

(b)

Estimating values for X:

Reward:

$$\mathsf{EstRwd(X)} = \sum_{P \in \boldsymbol{\mathcal{S}}} \mathsf{EstRwd}(P)$$

Cost:

$$\mathsf{Cost_1(X)} = \sum_{P \in \boldsymbol{\mathcal{S}}} \mathsf{EstCst}(P)$$

$$\mathsf{Cost_2(X)} = \frac{\sum_{(P,Q) \in C_2^{\boldsymbol{\mathcal{S}}}} \mathsf{MvCost}(P,Q)}{|C_2^{\boldsymbol{\mathcal{S}}}|}$$

$$\mathsf{EstCst(X)} = \mathsf{Cost_1(X)} + \mathsf{Cost_2(X)}|\mathbf{C_2^{\boldsymbol{\mathcal{S}}}}|$$

(c)

*$\boldsymbol{\mathcal{S}} = \{\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}\}$. $\mathsf{EstRwd}(i)$ and $\mathsf{EstCst}(i)$ are the estimated reward and cost values, respectively, for the point i. $\mathsf{MvCost}(i,j)$ is the cost of moving from point i to point j. $C_2^{\boldsymbol{\mathcal{S}}}$ is the set of pairwise combinations of the elements in $\boldsymbol{\mathcal{S}}$.

Figure 4.15: Estimating reward and costs for a more generic level of representation. (a) At the top is the level we want to estimate the values, while at the bottom is the level with the points that will be used as the basis for calculation. (b) Estimated values for the POIs at the more specific level. (c) Equations to estimate the reward and cost values at the more generic region.

Figure 4.15 shows the estimated values for the most generic region of an RSL, the representation at level one. We use the estimated values for the points in level two to calculate these values. Following the values in Figure 4.15a and applying the equation in Figure 4.15b, we have that the estimated reward for point X is 430.3, equal to the sum of the estimated rewards for points A, B, C, and D. We see that costs are composed of two values: $\mathsf{Cost_1}$ and $\mathsf{Cost_2}$. According to the values in Figure 4.15, $\mathsf{Cost_1}$ is worth 886.6, the summation of the estimated costs for each point at level two. $\mathsf{Cost_2}$ is computed using the cost average of moving between all combinations of two points at the more specific level. In the example, these combinations are $C_2^{\boldsymbol{\mathcal{S}}} = \{(\mathsf{A},\mathsf{B}), (\mathsf{A},\mathsf{C}), (\mathsf{A},\mathsf{D}), (\mathsf{B},\mathsf{C}), (\mathsf{B},\mathsf{D}), (\mathsf{C},\mathsf{D})\}$, and the computed average cost is 16.5. Thus, we estimate the cost of visiting X as 886.6 + 16.5 * 6 = 956.6, where 6 is the size of set $C_2^{\boldsymbol{\mathcal{S}}}$. We know this cost is overestimated, but it does not harm our planning method since the region-level planning just guides the second planning stage.

4.3.2 Region-level planning

After creating the RSL and estimating the costs and rewards of all regions at all levels, we can start planning the routes using the RegionLevelPlanning function. The idea at this step is to begin at the most general level and gradually assign agents to the most important areas of the representation, allowing more than one agent to visit the same point. We relaxed this restriction because a region represents a subset of POIs. So, in case of failure of any agent that passes through one important region, it is interesting that more than one agent is responsible for it. The only level where multiple visits to the same point are not allowed is the last one, where each region represents a single POI in the original environment. The Algorithm 3 shows the general idea of the region-level planning process.

Since a single region represents the entire environment at the first level, all agents will pass through the same point at that level. This is done in lines 2 through 4 of the algorithm. From the second level, we start directing agents to different areas, prioritizing the most valuable ones. Between planning one level and another, we use the routes generated in the previous level to select the set of POIs and to configure an initial route in the current level. This process, described in lines 7 to 17 of Algorithm 3, aims to reduce the search space in choosing the regions to be visited at a more detailed level and maintain coherence with the plan generated at the previous level.

In lines 9 and 10, we make a copy π'_i of each route generated in the previous level and reset the agent's route π_i to calculate a new one considering the current level. Each point in π'_i represents a set of points in the current level, so all these equivalent points are placed in the agent's set of interests. Furthermore, the getNearestProjections function also inserts in the interest set POIs from adjacent regions to avoid restricting too many possibilities. To set an initial route for the agent at the current level, the getFittestProjectedPoint function selects the most cost-effective POI within the region represented by point $v_j^{\pi'_i}$, which is on the agent's route at the previous level. Once this point is chosen, we add it to the route with the addPoint function on line 15. This function adds this point to the route at the lowest cost position.

At the end of line 16, each agent has an initial route through the regions it passed in the previous level and has a set ξ_i of POIs that are more attractive for each in the current level. From there, we perform a routes expansion process that consists of choosing the best POI to be placed in the solution, regardless of which route it will be inserted. This process is repeated until no points can be placed on any route. At each iteration of route expansion, the POI chosen to enter the solution is the one with the best cost-benefit ratio. In this computation, in addition to dividing the reward offered by the cost of visiting that POI, we consider each agent's free budget to balance the distribution of regions across

Algorithm 3: RegionLevelPlanning($RSL, \boldsymbol{\mathcal{A}}$)

```
1  //On the first level there is only one POI, so all agents go through it
2  foreach a_i ∈ 𝒜 do
3  |  π_i ← ⟨v_orig, 1, v_dest⟩
4  end
5  //Calculation of routes to the remaining levels
6  for k = 2 to |RSL| do
7  |  foreach a_i ∈ 𝒜 do
8  |  |  ξ_i ← { } // interest POIs to agent i
9  |  |  π'_i ← π_i
10 |  |  π_i ← ⟨v_orig, v_dest⟩
11 |  |  for j = 2 to |π'_i| do
12 |  |  |  ξ_i ← ξ_i ∪ getNearestProjections(RSL_{k-1}, v_{j-1}^{π'_i}, v_j^{π'_i})
13 |  |  |  p ← getFittestProjectedPoint(RSL_{k-1}, v_j^{π'_i})
14 |  |  |  π_i ← addPoint(π_i, p)
15 |  |  end
16 |  end
17 |  while it is possible to insert some point in some route do
18 |  |  c ← { }
19 |  |  foreach a_i ∈ 𝒜 do
20 |  |  |  c ← c ∪ [i, getFittest(ξ_i, π_i)]
21 |  |  end
22 |  |  p, a ← getMax(c)
23 |  |  π_a ← addPoint(π_a, p)
24 |  |  π_a ← optimize(π_a)
25 |  end
26 end
27 //Last level: we remove multiple visits to the same point.
28 foreach POI_x with more than one agent visiting it do
29 |  //selects the agent with the highest available budget
30 |  owner ← selectAgent(𝒜, POI_x)
31 |  foreach a_i ∈ 𝒜 − {owner} do
32 |  |  if POI_x ∈ π_i then
33 |  |  |  π_i ← removePoint(π_i, POI_x)
34 |  |  end
35 |  end
36 end
37 Φ ← {π_i, ∀a_i ∈ 𝒜}
38 return Φ
```

agents and prevent a single agent from being responsible for too many valuable points. Agents with a bigger available budget are likelier to visit a POI. The choice of the point and the route where it will be inserted is made in lines 18 to 22 of Algorithm 3, and in line 23, we insert that point in the route. In this planning step, we allow a region to be visited by more than one agent. To prevent all agents from taking the same route,

whenever an agent is allocated to visit a region, the addPoint function decreases the cost and reward that the region offers in half. If the region is much more important than others, it will continue being attractive even if it provides half the reward. We run a route optimization step at each new insertion to finalize the expansion process. The optimize function modifies the order in which the points are visited on the route, searching for the one with the lowest cost.

The whole process described in the previous paragraph is repeated until we reach the most specific level, where we will have a route for each agent in the POIs level of the original environment. However, due to the relaxation of the restrictions applied in this step, we can have different agents visiting the same point. To solve this problem, there is a route adjustment step where we remove multiple visits to the same point. This step is described between lines 28 and 35 of Algorithm 3. For each POI visited by more than one agent, we preserve it on the agent's route with the most available budget and remove it from the routes of the other agents. This choice aims to keep the routes more balanced concerning the agent's budgetary expenditure and free up the budget for them to use in the next planning stage. After ensuring no more points are visited by more than one agent, we have an initial route scratch at the POIs level. In line 37 of the algorithm, we create the generated routes set and return it in line 38 to be used in the second planning stage described in the next section.

4.3.3 POI-level planning

The process described in Section 4.3.2 gives a valid initial route for each agent containing POIs within regions that are important to visit. From these predefined routes, the POILevelPlanning function performs a series of insertions and removals to generate the final routes for each agent. The general idea of how the POILevelPlanning function works is described in Algorithm 4.

Since the last step of region-level planning was to remove multiple visits to the same POI, it is expected that the agents' budget is not yet fully committed. Thus, the first step in the POI-level planning is expanding each route to maximize each agent's reward collection, considering the COP. We consider the COP because, in this step, we plan the path for each agent as if it were alone in the environment without worrying about the rest of the team. The route expansion process is described in lines 1 to 9 on Algorithm 4, adding the most cost-benefit POI available until it is no longer possible to add points to the route due to a lack of budget. The setNeighborhood and updateNeighborhood functions create and update each agent's set of interest points according to its current

Algorithm 4: POILevelPlanning($\boldsymbol{\mathcal{V}}$, $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\Phi}$)

```
1  foreach a_i ∈ A do
2      ξ_i ← setNeighborhood(π_i)
3      while can insert points in the agent's route do
4          ξ_i ← updateNeighborhood(π_i)
5          p ← getFittestCOP(ξ_i, π_i)
6          π_i ← addPoint(π_i, p)
7          π_i ← optimize(π_i)
8      end
9  end
10 //remove multiple visits to the same point.
11 foreach POI_x with more than one agent visiting it do
12     //selects the agent with the highest available budget
13     owner ← selectAgent(A, POI_x)
14     foreach a_i ∈ A − {owner} do
15         if POI_x ∈ π_i then
16             π_i ← removePoint(π_i, POI_x)
17         end
18     end
19 end
20 // bad points removal step
21 foreach a_i ∈ A do
22     π_i ← removeExpensivePoints(π_i)
23     π_i ← removeDistantPoints(π_i)
24 end
25 // last stage of route expansion
26 while can add points to any of the routes do
27     foreach a_i ∈ A do
28         ξ_i ← updateNeighborhood(π_i)
29         p ← getFittestCTOP(ξ_i, π_i, Φ)
30         π_i ← addPoint(π_i, p)
31         π_i ← optimize(π_i)
32     end
33 end
34 return Φ
```

route. The getFittestCOP function searches among the points in ξ_i for the best cost-benefit, dividing each point's reward by the cost of inserting it in the route. Finally, the addPoint function adds the chosen point at the location with the lowest insertion cost within the route, and the optimize function seeks to improve the visit order of all points in the agent's route.

Ignoring teammates in this step generates the best possible individual routes from the initial ones, but we probably have the issue of multiple visits to the same point again. We then go through another process of removing points from routes. The method

described from lines 11 to 19 in Algorithm 4 works the same way as in Algorithm 3. The POI is kept on the agent's route with the most available budget. After that, in lines 22 and 23 of Algorithm 4, we have two more removal steps aimed at extracting POIs that are too costly to visit, or that leave the route with very dispersed points. The function REMOVEEXPENSIVEPOINTS removes the most expensive POIs to reach to prevent an agent from traveling too far to get a single POI. The function REMOVEDISTANTPOINTS removes POIs with the maximum average distances from all the other points on the route to keep the points closer together.

Following all these deletions, the agents have an available budget, and we perform the route expansion process for the last time. The previously POIs removed, if indispensable in the solution, will be placed in the route of some team member, which can even be on the same route as it was before the removals. The routes expansion step performed between lines 26 and 33 of Algorithm 4 is very similar to that performed at the beginning of the algorithm. The main difference is the use of the GETFITTESTCTOP function instead of GETFITTESTEOP. The function used here considers the routes of other agents to evaluate the correlation of information when choosing the best cost-effective POI to be inserted in each agent's route. Furthermore, in this expansion stage, different agents are no longer allowed to visit the same POI.

Once this whole process of inserting and removing points of routes is done, we have the mission planning to collaborate with the route adaptation procedure that will be used in case of failure of any team member. In the next section, we briefly analyze the computational cost of the Algorithm 4 and analyze the effectiveness of this method with the experiments in Section 4.3.5.

4.3.4 Computational Cost Analysis

The first step in our method is subdividing the POIs into regions and estimating the expected costs and rewards for these regions. The CREATEREGIONSPECIFICITYLEVELS and ESTIMATERSLVALUES functions of Algorithm 2 are responsible for this.

- CREATEREGIONSPECIFICITYLEVELS($\boldsymbol{\mathcal{V}}$): divides the POIs into regions and creates the Region Specificity Levels (RSL). First, it considers all POIs as a single region, and after that, at each new level of specificity, a region is divided into four others. So, each RSL will have $\log_4 |\boldsymbol{\mathcal{V}}|$ levels. We have to look at all the POIs at each level because we have to fit them in the correct region. So the complexity of this function is $O(\log_4 |\boldsymbol{\mathcal{V}}| \times |\boldsymbol{\mathcal{V}}|)$.

- EstimateRSLValues(RSL, $\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{A}}$): estimates the costs and rewards for the regions defined in the CreateRegionSpecificityLevels function. The biggest effort in this function is to calculate the average cost of one agent to move between all combinations of two points in a level. The way we create the RSL leaves the upper bound of the calculation of this function in the same order of generating all possible combinations at the most specific level of the RSL, namely $O(\frac{|\boldsymbol{\mathcal{V}}|!}{2\times(|\boldsymbol{\mathcal{V}}|-2)!})$.

After creating the RSL and estimating the values for the regions, we go through two planning stages, presented in Algorithms 3 and 4. As these two algorithms perform similar tasks, they share some functions. To simplify the analysis and avoid repetitions, we have grouped these functions according to the characteristic that most impact their execution time.

- Functions that depend on the number of agents in the team, that is, they have a complexity of $O(|\boldsymbol{\mathcal{A}}|)$.
  - getMax
  - selectAgent
- Functions that depend on the number of POIs. These functions, in the worst case, have complexity on the order $O(|\boldsymbol{\mathcal{V}}|)$.
  - getNearestProjections
  - getFittestProjectedPoint
- Functions that look at the neighborhood of a POI, that is, mainly depend on the agent's correlation range. The worst-case scenario for these functions is a correlation range that is so large that it covers all the POIs in the environment. The complexity in these cases is $O(|\boldsymbol{\mathcal{V}}|)$.
  - getFittest
  - setNeighborhood
  - updateNeighborhood
  - getFittestCOP
  - getFittestCTOP
- Functions that depend on the size of the agent's route, i.e. with complexity $O(|\pi|)$.
  - addPoint
  - removePoint
  - removeExpensivePoints

– removeDistantPoints

– optimize

Considering the lines of Algorithm 3, the greatest impact occurs between lines 6 and 26. The processes in lines 7 to 16 depend on the number of agents and the sizes of the routes. The worst-case scenario is when the agents can visit all the POIs, as the sum of all the routes would equal the size of set $\boldsymbol{\mathcal{V}}$. Therefore, the complexity of this snippet is given by $O(|\boldsymbol{\mathcal{V}}|^2)$. In lines 17 to 25, the steps are repeated as long as it is possible to insert points into a route. Since the worst-case scenario is to insert all the points, the complexity of this snippet is also $O(|\boldsymbol{\mathcal{V}}|^2)$. These two analyzed code sections are performed at each level of RSL, so the complexity between lines 6 and 26 is $O(\log_4 |\boldsymbol{\mathcal{V}}| \times |\boldsymbol{\mathcal{V}}|^2)$. From lines 28 to 36, the steps are repeated for all points with multiple visits. In the worst-case scenario, all the POIs are visited by two or more agents. In this case, the snippet in lines 31 to 35 will also analyze all the POIs. So, the complexity of the process between lines 28 and 36 to $O(|\boldsymbol{\mathcal{V}}|^2)$. After considering all the complexities mentioned, it can be concluded that Algorithm 3 has a worst-case complexity of $O(\log_4 |\boldsymbol{\mathcal{V}}| \times |\boldsymbol{\mathcal{V}}|^2)$.

In Algorithm 4, the section of lines 3 to 8 must be executed as long as it is possible to insert points in the agents' routes. The worst case is when all POIs can be entered. The complexity of the code section on lines 1 to 9 depends mainly on the size of $\boldsymbol{\mathcal{V}}$ set, as the team size is relatively small compared to the number of POIs. Thus, the worst-case scenario in lines 1 to 9 has a complexity of $O(|\boldsymbol{\mathcal{V}}|^2)$. Lines 11 to 19 describe the same process that lines 28 to 36 of Algorithm 3, so it has a complexity of $O(|\boldsymbol{\mathcal{V}}|^2)$. Functions between lines 21 and 24 depend on the size of the agents' routes. In the worst case, all POIs are visited, and the sum of points in all routes will equal the size of $\boldsymbol{\mathcal{V}}$. Therefore, the execution complexity of this snippet is $O(|\boldsymbol{\mathcal{V}}|)$. Finally, the code in lines 27 to 32 has a worst-case complexity of $O(|\boldsymbol{\mathcal{V}}|)$, but the process must be repeated as long as we can place points in a route. Assuming all the POIs will be visited, the worst-case scenario from lines 26 to 33 is $O(|\boldsymbol{\mathcal{V}}|^2)$. Adding up all the complexities, we can conclude that in the worst-case scenario, Algorithm 4 has a complexity of $O(|\boldsymbol{\mathcal{V}}|^2)$.

After considering all the complexities involved, we determined that the worst-case complexity of our method is $O(\log_4 |\boldsymbol{\mathcal{V}}| \times |\boldsymbol{\mathcal{V}}|^2)$. It is important to mention that this complexity is rarely reached since agents do not often have a budget to inspect all the POIs, and the correlation range of the agents usually does not cover all the POIs. Nevertheless, this analysis provides an estimate of the computational expenses required for our planning approach.

4.3.5 Experiments

This Section presents the experiments to evaluate our route planning methodology. We assess the quality of our method concerning the amount of reward planned for team collection and the impact of applying our method on monitoring mission simulations when one agent fails. We analyze the situations where no action is taken and those where we use the technique described in Section 4.2 to adapt the routes of active agents in the mission. The following subsections describe the experiments, obtained results, and conclusions in detail.

Conduction of the experiments

Section 4.1 mentioned the main methods to solve the CTOP in literature: the MIQP model [76] and the GA [66]. In Section 4.2.3, since our focus was to evaluate the proposed adaptation routine for the mission execution phase, we mainly used the GA to plan the missions because it is a more viable option to apply in a possible route replanning in cases of failure of one or more agents. However, this method has disadvantages, such as generating unbalanced routes and routes that underutilize each agent's available budget. In this Section, our goal is to evaluate the impact of the initial plan on the final reward collection when any agent fails in mission execution. Theoretically, using the MIQP method as a comparison parameter in these experiments would generate better routes for the team members.

We performed tests using the two cited planning methods to choose the fairest comparison parameter in analyzing the initial plans. As the GA is a heuristic that can vary the computed solution according to the execution, we generate ten plans for the same instance and choose that with the highest planned reward value as the initial plan for the team. Regarding the exact method, since calculating a solution to a MIQP problem can take minutes to days, depending on the instance, we set a maximum time of 36,000 seconds (10 hours) for the solver, accepting the best solution found in that time as the solution for the instance. In addition, we generate plans using our method described in Algorithm 2.

Once we had an initial route set, we performed the mission simulation following what was described in Section 4.2.3. At a certain point in the mission execution, an agent may fail, represented by excluding the agent from the simulation. However, in these experiments, we consider only the worst-case scenario, when an agent fails at the beginning of the mission. In other words, we simulate the case where the plan was made for a team with n members but executed by only $n-1$ of them. The agent who was taken out of the mission was always the one who would be responsible for the most profitable

route for the team. If the behavior of our method is positive in this scenario, it is expected that it will also be positive in more favorable ones. In the worst-case scenario, we consider the following strategies to evaluate the impact of the initial plan on the final result of the simulation:

- **Non-adapt**: the agent failure is ignored, and the remaining agents follow the initially planned routes;
- **Adapt**: applies the method described in Section 4.2 to adapt the routes of the agents that will execute the mission from the beginning.

The test environment consisted of a machine equipped with an Intel XEON E5-2650 CPU, 256GB of RAM, and Ubuntu 20.04. The MIQP problem was modeled in Python 3.5, and we used the Gurobi solver to calculate the solution. We used the GA provided by Tsiogkas and Lane, coded in C++. Our planning method was coded in Python 3.5.

Planning quality

The Gurobi solver ran until it found the optimal solution for the instances or until it reached the timeout of 36,000 seconds, returning the best solution found. As we do not have the guarantee that the optimal solution was always found, the first step was to evaluate the quality of these solutions. Gurobi returns a gap value along with the solution, representing the percentage difference between the solution found and the best upper bound obtained. A gap value equal to zero indicates that an optimal solution was obtained. Figure 4.16 shows the average of the gap values returned by the solver according to the instance size (number of POIs).

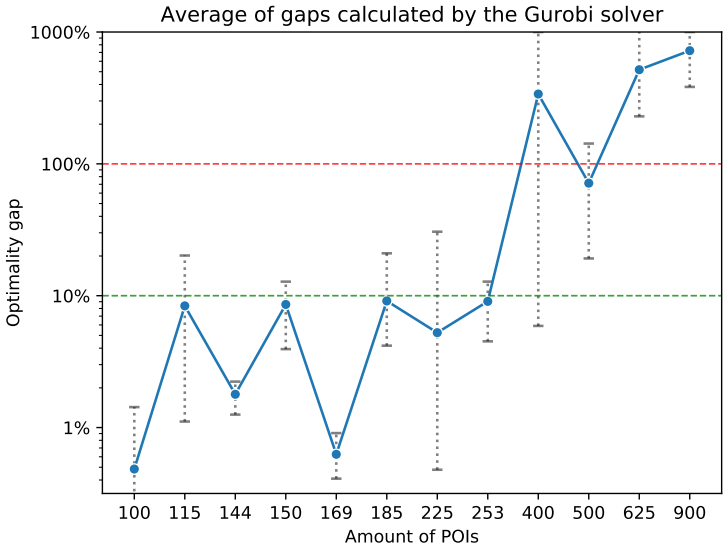
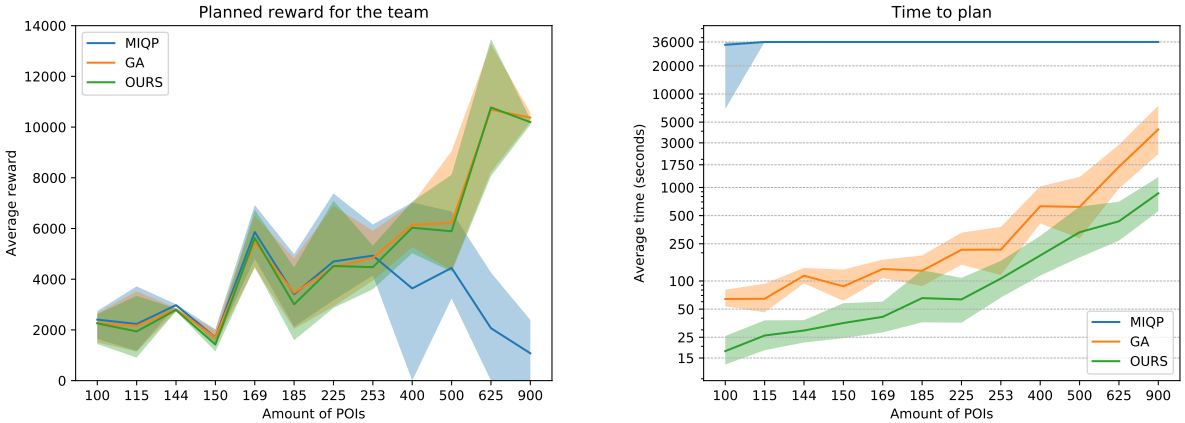


Figure 4.16: Optimality gap of the solution computed with Gurobi

We can see in Figure 4.16 that, for most instances, the solver could not compute an optimal solution within the established time limit. Only one instance with 100 POIs had the optimal solution computed. In Figure 4.16, we can also observe that for instances with 400 or more POIs, the average of the gaps returned by the Gurobi exceeds or is very close to 100%. This indicates that these solutions are too far from a good solution for the instance and that the solver would need more time to compute a better one.

Identified this issue of Gurobi with instances with a greater amount of POIs, we examine how our planning method and the GA behaved in these instances. Figure 4.17 compares the average planned rewards according to instances size using the three planning methods. We can see in Figure 4.17a that for instances with up to 253 POIs, the MIQP method is what provides the best solutions. However, if we consider instances with more than 400 POIs, where the exact method would need much more time than available, the two heuristics achieve a much higher average planned reward than the MIQP method. We also see in Figure 4.17a that our method has a slightly lower average planned reward than the other two methods. This was already expected since our method is a heuristic whose main planning objective is to reduce the impact of failures in the mission execution phase. The MIQP method and the GA, in turn, only aim to maximize the planned reward for the team.



(a) (b)

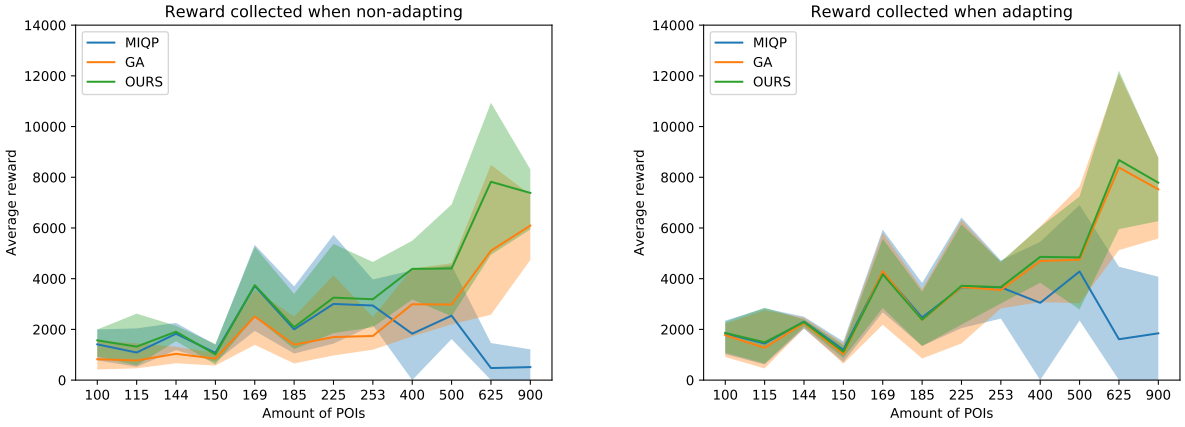
Figure 4.17: Comparison of initial plans obtained according to the number of POIs in the instance.

Another important factor to consider, in addition to the quality of the solution, is the time spent calculating it. We can see in Figure 4.17b that the MIQP method has a high running average time, mostly caused by the timeout. The only instance that did not cause the timeout was the one where Gurobi was able to calculate the optimal solution, which even so was solved in more than 5,000 seconds. We run the GA ten times to determine an initial plan. So, the time spent to solve an instance with this method is the sum of the ten executions. Even so, the average time spent by the GA is much

lower than that spent by the MIQP method. Finally, we have our method as the one that takes the least amount of time on average to get an initial plan for a team. Our approach spends, on average, half the time required to run the GA ten times and is up to 36 times faster than the MIQP method.

Simulating missions with agent failures

Having inspected the quality of the generated plans, we move on to the mission execution phase, where an agent fails right at the beginning, and the team will carry out the entire mission with one component less. We already know that our method is faster than MIQP and GA. Regarding the reward, our method plans a slightly smaller amount than the other two. In this Section, we want to verify if our approach is effective in situations of agent failure. Figure 4.18 compares the team's final reward collection when using the three methods to generate the initial route plan.



(a) (b)

Figure 4.18: Comparison of final reward collected when one agent fails according to the number of POIs in the instance.

We can see in Figure 4.18a that in cases where the team members ignore the agent failure and follow the initially planned routes, our method provides an average reward collected equal to or greater than the missions that used the plan generated by the MIQP method. About the GA, our superiority is much more evident. This large difference is mainly due to the unbalanced routes that GA generates, which we discussed in Section 4.2.3. These results indicate that our planning approach, even if the agents do nothing about the teammate's failure, is better at distributing the agents among the regions most valuable in the environment. Figure 4.18b shows the average rewards collected in missions applying the adaptation routine described in Section 4.2. We can see that, except for the MIQP method in instances with more than 400 POIs, the average final reward collected was closer regardless of the planning method used. This result is

due to a great influence on our adaptation routine, which manages to recover a good part of the reward that would be lost due to the agent's failure.

To better understand the impact of our planning method on the final result of the mission, we directly compared the solutions computed using the MIQP method with the solutions calculated by our method. We chose this method for this direct comparison due to the results shown in Figure 4.18a, which indicate that the impact of a failure on the plane generated by GA is much greater than on a plane generated by the MIQP method. To be fairer in the comparison, we only considered the instances where Gurobi solved the problem with an average gap below 10%. In the analyses shown in Figure 4.19, we only considered instances with a maximum size of 253 POIs.

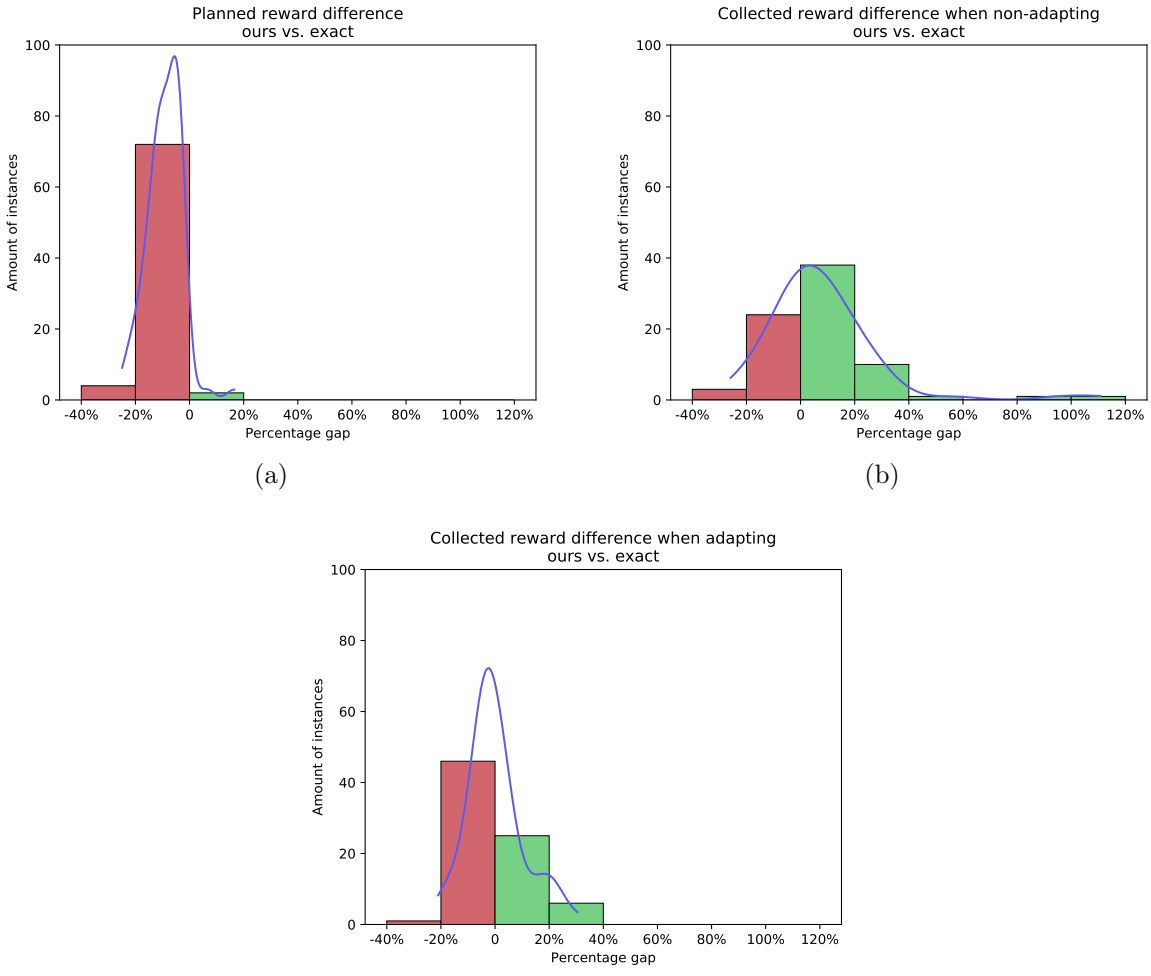


Figure 4.19: Direct comparison between instances using MIQP against our planning method. The green bars count the instances in our method that had an advantage over the MIQP method, while the red bars indicate the opposite.

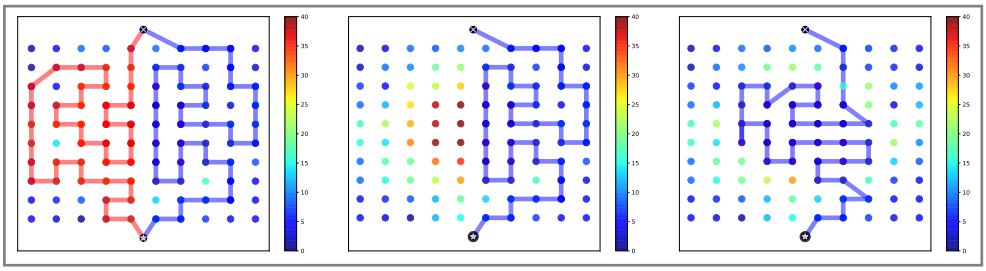
We see in Figure 4.19a that the vast majority of plans generated using the exact method estimate a reward of up to 20% more than what our method was able to plan. This was expected, as we are not only interested in maximizing the planned reward for

the start of the mission as the exact method is. Even so, looking at the density curve plotted on the histogram, we can see that most of our solutions plan rewards about 10% less than what was planned by the exact method considering the same instance.

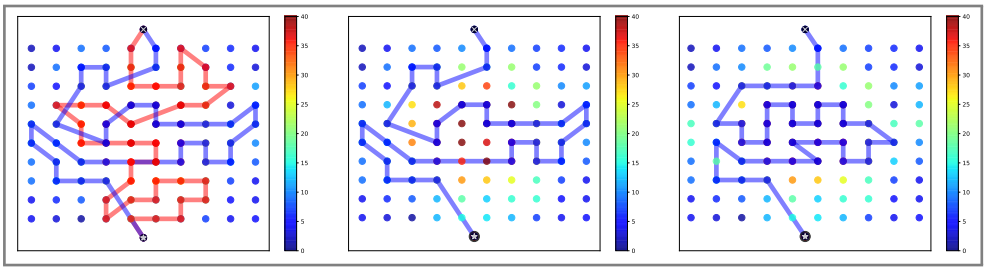
Our planning method aims to generate plans prepared for agent failures during mission execution rather than only maximizing the planned reward for the team. This concern with failure is clear in the results shown in Figure 4.19b. We can see that when an agent fails and others ignore this event, our planning method generates routes that lead the team to collect more rewards at the end of the mission. This means that our approach can better divide the most important regions of the environment among all team members. Figure 4.19c compares the rewards collected when we applied the online adaptation of the remaining agents' routes in the mission. This plot shows that we cannot always recover more rewards using our planning strategy than the routes calculated by the MIQP method. However, in most cases where our planning method had a lower performance than the MIQP, the final reward collected is below a 10% difference. In the cases where our planning method had a higher gain, the difference is more expressive. Furthermore, considering the time spent generating the initial plan, our planning method is more cost-effective.

We also performed a qualitative analysis of the generated routes. One worth-mentioning case is that where the MIQP method found the optimal solution for the instance. Figure 4.20 compares the routes generated for this instance using the MIQP and our planning method. The environment consists of 100 POIs arranged in a grid and a region more important to inspect in the central area. In Figure 4.20, the top line shows the results obtained with MIQP, and the bottom is the results of executing our method. The first image of each line represents the initial planning generated. The other two images in each line represent the scenario in which the agent responsible for the most valuable route fails right at the beginning of the mission. In the situation illustrated in the central image, the agent following the mission ignores the failure and follows the path initially planned. The last image of each line is the result of the simulation when the agent responsible for the blue route adapts its path to minimize the teammate's failure.

The first image of each line in Figure 4.20 clearly shows the difference in the planes generated by the two methods. Our method intertwines the two routes in the central region of the environment. In contrast, the routes generated by MIQP are disjoint, making each agent responsible for inspecting half of the environment. Our grid instances have POIs equally spaced vertically and horizontally. This means that displacements in these directions cost less than diagonal displacements. Thus, our plan may seem counter-intuitive if we ponder the reward collected per distance covered. Our routes have many more diagonal displacements than the routes generated by MIQP and consequently spend more of the agents' budget, which decreases the number of POIs that will be visited. However, our concern is not only to maximize the reward of the plan but that agents'



(a)



(b)

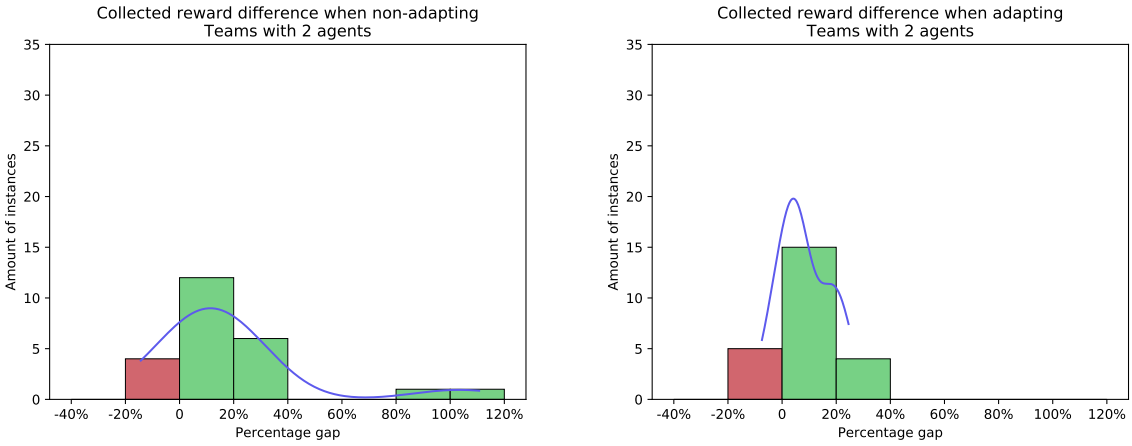
Figure 4.20: Instance with optimal solution calculated by the MIQP method. (a) results obtained by running MIQP; (b) results obtained with the execution of our planning method. From left to right, the plotted routes represent 1. initial plan; 2. failure ignored by the teammate; 3. The active agent adapts its route to mitigate the failure.

failure in the mission execution phase does not have a very big impact. Thinking about this issue, the intertwined routes start to make more sense because depending on the point of failure of an agent, the other can better adapt its courses to cover important regions that would be neglected. Even in cases where the failure is ignored, a plan like ours is more beneficial, as it ensures that at least one of the agents will pass through the most important region of the environment and acquire some knowledge about it.

During the qualitative analyses, we realized that the number of elements in the team affects the amount of reward recovered at the end of the mission when using our method or the MIQP method to generate initial routes. This phenomenon mainly occurs because the MIQP method better spreads the agents' routes throughout the environment without worrying about failures or regions with different importance to inspect. Since more routes are spread out in the environment, more than one will probably pass through one important region, reducing the impact of a failure and facilitating the route adaptation routine. When we have few routes, they tend to become more disjointed in the environment, as shown in Figure 4.20.

We separated a set with 24 different environment representations and planned routes for teams with 2, 3, and 4 agents for these environments. In Figure 4.21, we see the difference in the percentage of the final reward collected by a team of 2 agents. It is clear from the two histograms that using our method to generate the initial routes was more advantageous when the teammate ignored the failure of the other agent and when it had its route adapted during the continuation of the mission. Our method performed

better in scenarios considering teams with two agents than with other team sizes. This is because we intertwine routes in regions with greater inspection importance, while the MIQP method generally makes only one agent responsible for the region.

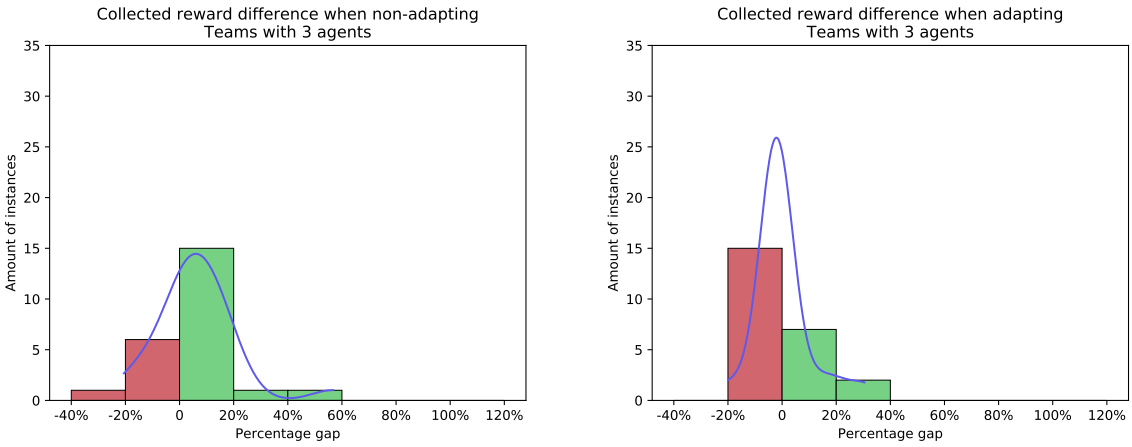


(a) (b)

Figure 4.21: Direct comparison between instances using MIQP against our planning method considering only teams with two agents. The green bars count the instances in our method that had an advantage over the MIQP method, while the red bars indicate the opposite

Figure 4.22 shows the comparative result for instances where the disposition and rewards of the POIs are the same used with two agents but now considering three agents in the team. We can see in Figure 4.22a that using our planning method still has a certain advantage in situations where the remaining agents ignore the teammate's failure. In Figure 4.22b, we see that when other agents adapt their routes, using the exact method as the planning strategy becomes more advantageous than in cases where we plan for teams with two agents. By having one more route than in the previous scenario, the MIQP method makes one agent responsible for a large part of a most important region, and at least one of the other two routes will pass close to that region. By passing a route close to the most valuable POIs, the adaptation routine can better adapt the routes to cover part of the region with greater importance to inspect. Even so, looking at the difference in percent gain, we see instances with substantially greater increases when using our method to plan the initial routes.

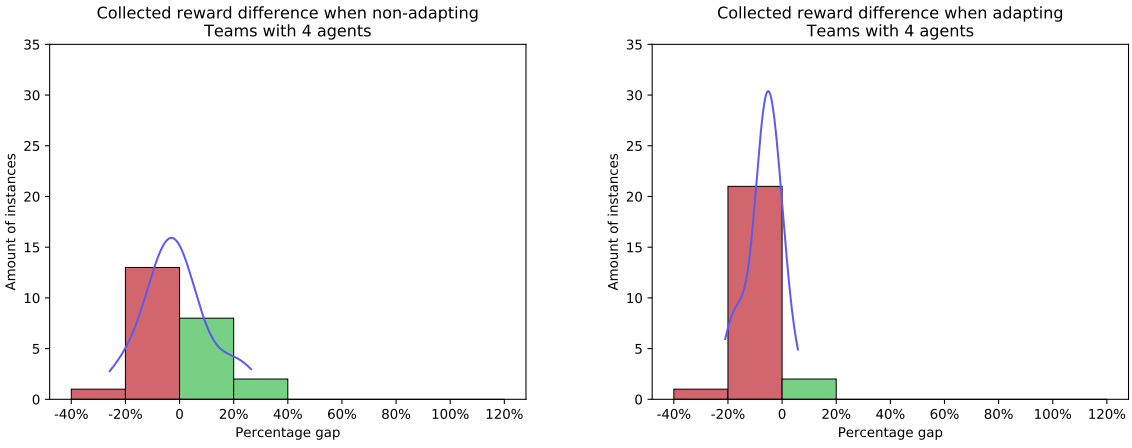
In Figure 4.23, we see the result for the same environment representations but now consider teams with four agents. Figure 4.23a indicates that when the teammates ignore the failure, it is still worthwhile to use our planning method in some cases. However, the MIQP method is already starting to be more advantageous regarding the final reward collected since, with the increase in the number of routes in the environment, possibly more than one route is passing through a region with greater inspection importance. In Figure 4.23b, we see that using the MIQP method to plan the mission along with our



(a) (b)

Figure 4.22: Direct comparison between instances using MIQP against our planning method considering only teams with three agents. The green bars count the instances in our method that had an advantage over the MIQP method, while the red bars indicate the opposite

adaptation routine when the failure is noticed is where we collect the highest reward at the end of the mission execution. In this case, using our planning method manages to collect more rewards at the end of the mission in less than five instances only. Even so, we can see from the density distribution curves that for instances where using the MIQP method results in a higher reward collection, this value is, in most cases, about 10% more reward than when we use our method to plan the routes for the same instances.



(a) (b)

Figure 4.23: Direct comparison between instances using MIQP against our planning method considering only teams with four agents. The green bars count the instances in our method that had an advantage over the MIQP method, while the red bars indicate the opposite

4.4 Concluding remarks

In this Chapter, we presented several experiments that we performed to show the behavior of our method in different situations. We noted that the success of our approach depends on some factors in addition to the parameters considered in the algorithm. These factors are linked to the initial planning and at what time of the mission any agent fails.

We designed our adapting heuristic to make minor adjustments to the routes to cover important regions abandoned by other agents without completely giving up the initial planning. Therefore, it is desired that more than one route pass close to the areas with more valuable POIs so that, in the event of any agent's failure, another agent can be covered the region without much effort. Otherwise, many POIs must be removed from the agent's route to increase the available budget to reach the region. Moreover, the sum of rewards for these removed POIs is probably more significant than the reward for the POI reached in the region, thus not compensating for the change in the route.

Another point is the agents' budget limit and their position in the route when a failure occurs in one of the teammates. Suppose an agent nearing the end of its route realizes that another team member will no longer visit a valuable region. In that case, the agent will probably not have a budget to adjust their route and visit that region. Thus, some high-reward POIs may not be visited due to a lack of budget.

Since initial planning is an impacting factor in mission execution when failures occur, we propose a route planning method that is prepared to deal with these situations. Our heuristic method divides the environment into regions and seeks to distribute the agents to these regions keeping more than one agent in the most valuable ones. Therefore, if one of these agents fails, the area will not be completely uncovered. As we do not focus exclusively on maximizing the planned reward but on that one collected at the end of the mission execution when there are agents failures, the plans generated by our heuristic estimate a total reward lower than the planned using the MIQP method and the GA proposed by Tsiogkas and Lane.

Analyzing the reward collected by the team at the end of the mission execution, our planning approach proved to be quite effective, especially when agents ignore the failure of a teammate and follow the routes present in the initial plan. The advantage over the results obtained using the GA is even more expressive due to the problem of unbalanced routes of this algorithm. In cases where the remaining agents in the mission adapt their routes, our planning method produces plans with a final reward collected very close to the MIQP method and the GA, with a slight advantage of MIQP in instances with up to 253 POIs. For instances with a larger number of POIs, where the MIQP method cannot compute a good plan, our planning method proved to be more effective than GA in producing plans with a greater reward collection when occurs a failure of one agent

during the mission execution. Despite having a worse performance in some instances, which is around 10% less reward on average, our planning method proved to be faster than the other two to generate a solution, which may justify its use.

In a direct comparison with the MIQP method, besides the time spent to generate a solution, our method takes advantage when we have teams with few agents. This is because, when we have few routes, the MIQP method tends to place them disjoint in the environment, preventing the routes are too close to each other to maximize the collected reward, which impairs the proposed adaptation routine. As we increase the number of team members, even aiming only to maximize the collected reward, the MIQP method better spreads the routes across the environment, making more than one agent pass through the most valuable regions. Therefore, with more agents, the MIQP method has the advantage over our planning heuristic. But we saw that this advantage is about 10% more on collected reward with a planning cost 36 times higher concerning time.

With all that was exposed, we can say that using the planning method proposed by us is a great advantage even if the team members ignore the failure of a partner. Besides, along with the proposed route adaptation method, we have slightly lower results regarding the final collected reward than when we use the MIQP method or the GA. But we greatly improve the time spent to generate the plans compared with the MIQP method, and we prevent the unbalanced routes issue of the GA. These facts make our planning strategy a great option for planning missions that failure-prone agents will execute.

Chapter 5

Heterogeneous teams

In previous chapters, we explored the problem of monitoring areas with different inspection importance using teams with homogeneous agents. This approach allowed us to verify the effectiveness of our methods regarding the methods described in the literature, which in great majority use teams composed of agents with the same characteristics. However, considering a real-world monitoring mission performed by a team of autonomous agents, it is more plausible that agents have different characteristics. In this chapter, we consider autonomous agents with different budgets, costs, and perception capabilities, working as a team to monitor the same environment, thinking that one or more may fail at some point in the mission execution phase.

This Chapter is divided into three Sections. Section 5.1 describes the heterogeneity of agents, giving examples of autonomous robots and the different characteristics we consider in our approach. Section 5.2 explains the changes in our planning method to deal with heterogeneous agents and describes the experiments and the results obtained in this scenario. Finally, Section 5.3 brings the concluding remarks on this topic.

5.1 Agents heterogeneity

An area monitoring mission can employ a wide range of autonomous agents. The use of agents with different characteristics can increase monitoring efficiency and quality. For example, some agents may be designed to traverse rough terrain, while others may have a more powerful battery for more extensive monitoring. Several characteristics, such as mobility, attached sensors, and energy source, may vary from one agent to another. In this dissertation, we chose three elements to explore: uptime autonomy, moving and sensing costs, and the capability to perceive the environment. These characteristics interfere directly with our problem modeling. Uptime autonomy concerns the available budget for each agent, that is, the available time to carry out its route. The moving cost is associated with the speed at which the agent moves in the environment, while the sensing cost

is equivalent to the time it takes to collect information at the POI. Finally, the agent's perception capability is related to the sensors that it has and the quality of these sensors.

Without loss of generality, and to better illustrate the scenario of failures in heterogeneous teams, we consider two categories of autonomous agents: aerial and ground robots. In addition to facilitating the understanding of what we proposed, these two categories were chosen because they are two types of agents widely familiar and used in area monitoring tasks. Sections 5.1.1 and 5.1.2 describe each of these two categories of robots, emphasizing the characteristics we are interested in, while Section 5.1.3 focuses on how robots can perceive the environment. Finally, Section 5.1.4 summarizes the attributes of the robots and sensors, showing how we composed the heterogeneous teams.

5.1.1 Aerial robots

Autonomous aerial robots can fly over large land extensions and perform tasks faster and more efficiently than other agents. In this dissertation, we chose to highlight drones among the various classifications of existing aerial robots because they are widely familiar and used in area monitoring missions. Figure 5.1 shows some models of drones with different propulsion systems that are commonly used in monitoring tasks. In addition, drones also differ in size, speed, flight time range, and the sensors and equipment they are equipped with, which are the components we are interested in.



(a) eBee X (b) FX10 (c) Aeromapper Talon

(d) EVO II Pro (e) Typhoon H Plus (f) Falcon 8+

Figure 5.1: Example of commercial drones. Images are not on the same scale.

Figures 5.1d, 5.1e, and 5.1f show multi-rotor drones. These drones are designed to take off and land vertically and are more versatile for indoor or hard-to-reach areas. They usually have 4, 6, 8, or more rotors, which ensure they maintain their position and stability in the air with greater precision and are easily maneuverable. In contrast, the flight time range of these drones is limited by the need to keep several engines running. In Table 5.1, we show commercial multi-rotor drone models with information such as the number of rotors, flight time autonomy, and maximum speed reached by each one.

Table 5.1: Some commercial multi-rotor drone models.

Model	**Manufacturer**	**Number of rotors**	**Uptime Autonomy**	**Maximum Speed**
EVO II Pro	Autel Robotics	4	Up to 40 min	20.0 m/s
Falcon 8+	Intel Corporation	8	Up to 26 min	18.0 m/s
IF1200	Inspired Flight	6	Up to 24 min	22.0 m/s
Phantom 4 Pro	DJI	4	Up to 30 min	20.0 m/s
PowerEye	PowerVision	4	Up to 29 min	18.0 m/s
Typhoon H Plus	Yuneec	6	Up to 28 min	13.4 m/s
X-Star Premium	Autel Robotics	4	Up to 25 min	15.0 m/s

Figures 5.1a, 5.1b and 5.1c show fixed-wing drones. These drones need to take off and land on flat surfaces and can fly at higher altitudes. They are more efficient than multi-rotor drones in speed and flight time range, but they are less maneuverable and cannot stay stationary in the air. In Table 5.2, we show commercial fixed-wing drone models with their flight time autonomy and maximum speed reached.

Table 5.2: Some fixed-wing drone models.

Model	**Manufacturer**	**Uptime Autonomy**	**Maximum Speed**
Aeromapper Talon	Aeromao	Up to 120 min	30.5 m/s
eBee X	senseFly	Up to 90 min	30.5 m/s
PHX	Sentera	Up to 59 min	24.4 m/s
Sitaria E	Uavos	Up to 180 min	38.8 m/s
SkyRobot FX10	Robot Aviation	Up to 120 min	40.0 m/s
X8 Flying Wing	Airelectronics	Up to 180 min	30.5 m/s

It is worth noting that the information presented in Tables 5.1 and 5.2 are approximate. However, we can observe that flight time range and speed vary between the different types of drones. Comparing these two types of drones shows that multi-rotor generally have a lower flight range and speed than fixed-wing drones. Also, fixed-wing drones can

reach a higher altitude than multi-rotors, which can benefit some tasks. However, this difference in characteristics makes fixed-wing drones considerably more expensive than multi-rotor ones.

5.1.2 Ground robots

Autonomous ground robots move and operate on the soil. Figure 5.2 shows some ground robot models used in area monitoring tasks. The clearest difference when looking at these robots is the locomotion system employed. Ground robots commonly use wheels, legs, or crawler tracks to move in an area monitoring mission. The choice of the robot's locomotion system should be directly linked to the characteristics of the terrain where it will operate. Wheeled robots are generally faster at moving on flat surfaces, while legged ones can move over rough terrain and adapt to different types of surfaces. Tracked robots, on the other hand, move well on flat or hilly terrain and provide greater safety on sandy or poorly gripped soil.



(a) Bunker Pro (b) Caliber T5 (c) Digit



(d) Spot (e) Summit-XL (f) Warthog

Figure 5.2: Example of commercial ground robots. Images are not on the same scale.

Ground robots generally have a higher uptime autonomy than aerial robots. However, robots that move with legs may have slightly lower operation time than those that use bearings due to the additional effort of moving the legs correctly to maintain balance. Regarding locomotion speed, robots with legs are also at a disadvantage compared to the other two. The fastest robots of the three are those with wheels. In Table 5.3, we

list some commercial models of ground robots that can be used in monitoring missions, classifying them by locomotion system and with information on uptime autonomy and maximum speed achieved.

Table 5.3: Some ground robot models

Model	**Manufacturer**	**Locomotion system**	**Uptime Autonomy**	**Maximum Speed**
AlienGo	Unitree Robotics	legged	Up to 3.5h	1.6 m/s
Digit	Agility Robotics	legged	Up to 3h	1.5 m/s
Spot	BostonDynamics	legged	Up to 1.5h	1.6 m/s
Bunker Pro	AgileX Robotics	tracked	Up to 3h	1.5 m/s
Caliber T5	ICOR Technology	tracked	Up to 4h	1.9 m/s
TrackReitar	Leotronics	tracked	Up to 7h	2.7 m/s
Probot	Roboteam	wheeled	Up to 8h	3.3 m/s
Summit-XL	Robotnik	wheeled	Up to 5h	3.0 m/s
Warthog	Clearpath	wheeled	Up to 2.5h	5.0 m/s

It is important to note that the values presented in Table 5.3 are approximate and may vary according to the operating conditions of the robots. Compared to aerial robots, we can notice a higher uptime autonomy of ground robots but a lower average locomotion speed. The biggest limitations of ground robots as those imposed by the terrain, which can cause jams or skids that make locomotion in the environment unfeasible. Regarding the equipment cost, ground robots are generally cheaper to produce and maintain than aerial robots.

5.1.3 Robots perception

Autonomous robots can navigate, perceive, and interact with the environment without human interference. In this dissertation, we are interested in the robot's ability to perceive and collect data from the environment, a fundamental task in area monitoring missions. The robots collect information from the environment through their sensors, ranging from cameras that capture images and video to sensors for environmental measurements such as temperature and humidity. Besides the type of information they collect, sensors can be very different in terms of technology and performance, with varying capabilities of measurement accuracy. Overall, the perception capability can be improved by

adding more sensors and equipment to the robot, but this can also increase the cost and complexity of the system.

The data quality captured by a sensor depends on several aspects. For example, consider that a robot is equipped with an Red, Green and Blue (RGB) camera. The information quality obtained in this case will depend on sensor configurations such as resolution, lens, focus, and factors related to the operating conditions, such as luminance, smoke, possible robot vibrations, and so on. In this dissertation, without loss of generality, we ignore external factors of the sensor and focus on two properties directly related to the CTOP: the quality of the information collected and the sensor range at a given POI. In CTOP, these values are linked to the amount of reward collected at each POI and the neighborhood size considered to compute the correlated reward, respectively. These two values are inversely proportional; that is, while one increases, the other decreases. To illustrate this concept, consider that an aerial robot carrying a RGB camera captures an image over a POI at 500 meters of height. This image has a determined spatial resolution and covers a certain area around the POI. In contrast, suppose the robot flies over the same point at 1,500 meters of altitude and captures an image in the same coordinates. In that case, the spatial resolution of this image is lower, and it contains much more information about what is around the point.

The explanation in the previous paragraph used a RGB camera because it is a widely disseminated sensor. However, various sensors can be attached to aerial or ground robots to collect information, such as thermal or multispectral cameras, gas detectors, humidity sensors, etc. Regardless of the sensor and the type of robot that carries it, the concept explained above is still valid: the more sensitive and precise a sensor measures a stimulus from the environment, the less information about its surroundings we can infer. In contrast, a less accurate sensor can get information about a larger area with just one measurement, but the quality of this measurement is inferior. In the next Section, we describe how we modeled the different types of robots and sensors we considered when assembling our heterogeneous teams for monitoring areas.

5.1.4 Robots teams

Since robots have specific capabilities and limitations, using different types of robots together allows for obtaining data and information from diverse sources and points of view, improving mission performance. In this dissertation, we are interested in studying the behavior of a heterogeneous team when one or more robots fail during the area monitoring mission and cannot inspect the region assigned to it. For example, when we

have a set of robots with different maximum speeds, should the fastest or the slowest robot be responsible for the most important regions of the environment? What would be better, considering that some team element might fail and that the other robots will adapt their routes to cover part of an important region that would be neglected?

To answer questions like these, we first had to define the set of robots we would use in these analyses. Based on what was described in Sections 5.1.1 and 5.1.2, we generalized the robots into five categories according to the characteristics we were interested in, as shown in Table 5.4. The idea of this generalization was to group the robots by their uptime autonomy and maximum speed without the need to quantify this information. In Table 5.4, it is clear that ground robots have a longer autonomy than aerial robots, which are much faster. For example, while multi-rotor aerial robots have the shortest uptime autonomy, they are more than twice as fast as wheeled ground robots, which have the longest uptime autonomy among those considered in the study.

Table 5.4: Classification of robots used in the analysis of heterogeneous teams.

Category	**Robot type**	**Locomotion system**	**Uptime autonomy**	**Maximum speed**
Aerial Multi-Rotor (A-MR)	aerial	multi-rotor	◐○○○	●●●○
Aerial Fixed-Wing (A-FW)	aerial	fixed-wing	●●○○	●●●●
Ground Robot Legged (G-RL)	ground	legged	●●◐○	◐○○○
Ground Robot Tracked (G-RT)	ground	tracked	●●●◐	●○○○
Ground Robot Wheeled (G-RW)	ground	wheeled	●●●●	●●○○

In addition to the robots, a wide variety of sensors can be used to collect data from the environment. Since we are only interested in the quality of the collected information and the sensor range, we show in Table 5.5 the generic sensors we considered in the experiments. It is worth noting that these sensors can be attached to any of the robots without loss of generality.

Table 5.5: Classification of sensors used in the analysis of heterogeneous teams.

Classification	**Information quality**	**Correlation range**
Bad Quality - Higher Range (BQ-HR)	◐○○○	●●●●
Poor Quality - Wide Range (PQ-WR)	●○○○	●●●○
Medium Quality - Medium Range (MQ-MR)	●●○○	●●○○
Good Quality - Small Range (GQ-SR)	●●●○	●◐○○
Highest Quality - Lowest Range (HQ-LR)	●●●●	●○○○

Once we created classes of robots and sensors, we would need a criterion for assembling teams with these elements. To maintain a reasonable basis for comparison, in all teams, we use at least one robot with the same characteristics as the agents used in Chapter 4, which we call Standard Robot (SR). Thus, we would have a way of comparing the difference between homogeneous and heterogeneous teams in the monitoring mission for the same environment.

In the experiments reported in Section 5.2, we classified SR into one of the categories listed in Table 5.4. The other robots that comprise the team vary according to the property evaluated each time and respecting the categories defined in Table 5.4. For example, if we are analyzing the heterogeneity concerning the maximum speed of the robots and the SR is classified in the G-RW category, we would have another robot in the A-MR category, which has twice the maximum speed or a robot in the G-RL category with half the top speed. The robots considered in each experiment are detailed in the corresponding Sections.

5.2 Mission planning and execution

Intuitively, we kept the same planning strategy described in Section 4.3. If dividing the environment into sub-regions worked well for agents with the same characteristics, it is expected to work just as well when agents have different ones. However, some adaptations in the method had to be implemented to deal with this new scenario. First, when considering homogeneous teams, all agents view the environment similarly. That is, the rewards and costs of moving around and collecting rewards are the same for all agents. When the agents are different, we know precisely these values will vary from one to another. Therefore, instead of creating a single Region Specificity Levels (RSL) shared among all team members, we make an RSL for each agent when dealing with heterogeneous teams. All RSL have the same structure of levels and divisions of regions. However, there is a difference in the values of rewards and estimated costs in each region, which are now calculated according to the characteristics of each agent. When estimating costs, we considered the agents' speed and the time spent to sense one point. To calculate the expected rewards, we used the data quality captured by the sensor and the correlation range of each one.

In addition, two processes in our method can directly influence when the agents have different characteristics. The first occurs in region-level planning, specifically when we set an initial route for the agent in an RSL level, using the route calculated in the previous level. This process is described in lines 10 to 15 of Algorithm 3. Since in region-

level planning, we allow multiple visits to the same POI, and the process in question is done sequentially for the team members, the order in which we choose the agents to make the routes may favor one element or another. When the agents had the same characteristics, there were not many problems with this choice. However, if the agents had different speeds, it would be interesting to know if the fastest or the slowest should have priority to visit a POI that is of interest to both, as this choice will probably cause impacts in possible failures of an agent during the mission execution.

In the same line of thought, the other process that deserves much attention is removing multiple visits to the same POI. This happens at the end of Algorithm 3 (lines 28 to 36) and also in Algorithm 4 (lines 11 to 19). Due to the great impact that this process can have on our method, we highlight it in Algorithm 5. We see in line 4 that the SELECTAGENT function is responsible for determining which team agent has priority in keeping a POI that interests more than one team member. In the version for homogeneous teams, we always choose the agent with the highest available budget. The SELECTAGENT function subtracts the route cost of the agent from its total budget and chooses the agent with the highest value in this difference. However, if we were interested in analyzing another characteristic, such as perception capacity, for example, the available budget would probably not be a decisive factor in determining which agent has priority in having a certain POI on its route.

Algorithm 5: RemoveMultipleVisits($\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{A}}, \boldsymbol{\Phi}$)

```
1  //remove multiple visits to the same point.
2  foreach POI_x with more than one agent visiting it do
3      //selects the agent with the highest available budget
4      owner ← SELECTAGENT(A, POI_x)
5      foreach a_i ∈ A − {owner} do
6          if POI_x ∈ π_i then
7              π_i ← REMOVEPOINT(π_i, POI_x)
8          end
9      end
10 end
```

It was also necessary to implement changes in the route adaptation routine proposed in the previous Chapter. However, these changes were only to ensure that heterogeneities would be considered for each agent when adapting the routes. This was necessary because, previously, all team agents had the same costs to perform tasks and perceived the environment similarly. In the experiments of this Chapter, the budget and neighborhood range parameters of the adaptation routine depend on each agent. We consider that the neighborhood range is equivalent to the correlation range of the robot. In addition, we defined the size of the analyzed segment as $la = 20$ and the number of nodes to be removed as $rem = 15\%$ of these nodes to have results consistent with those in Chapter 4.

To better understand the impact of the heterogeneity analyzed in this study, we treated each characteristic separately, reducing the interference of other factors. According to the evaluated heterogeneity, we modified the agent selection order to set an initial route and the criterion used by SELECTAGENT function for defining the agent with higher priority to visit a POI. For clarity, we describe separately the modifications made to the planning method and the experiments and results of each of the analyses made in Sections 5.2.1 to 5.2.3. Section 5.2.1 deals with heterogeneity regarding available budgets to comply with the mission. Section 5.2.2 treats agents with varying perception capabilities. Finally, Section 5.2.3 considers agents with different movement and sensing costs.

5.2.1 Heterogeneous budgets

The first heterogeneity analyzed was the budget available for the agent to carry out the mission, representing the time the robot can operate uninterruptedly before exhausting its power source. In this analysis, we tried to understand the impact of having robots with different uptime autonomy in the same mission and how our method would behave when generating routes according to the robot's ability to operate for more or less time. Without loss of generality, we assume all robots have the same maximum speed and are equipped with the same sensor to get information about the environment. Thus, the only feature that differentiates robots is uptime autonomy. Table 5.6 summarizes all robot types considered in this experiment.

Table 5.6: Considered robots in the experiments with heterogeneous budgets. The classification was given exclusively about uptime autonomy.

Robot type ID	Robot class	Uptime autonomy
WINGS*	Aerial Fixed-Wing (A-FW)	●●○○
LEGS	Ground Robot Legged (G-RL)	●●◐○
TRACKS	Ground Robot Tracked (G-RT)	●●●○
WHEELS	Ground Robot Wheeled (G-RW)	●●●●

* Standard Robot configuration.

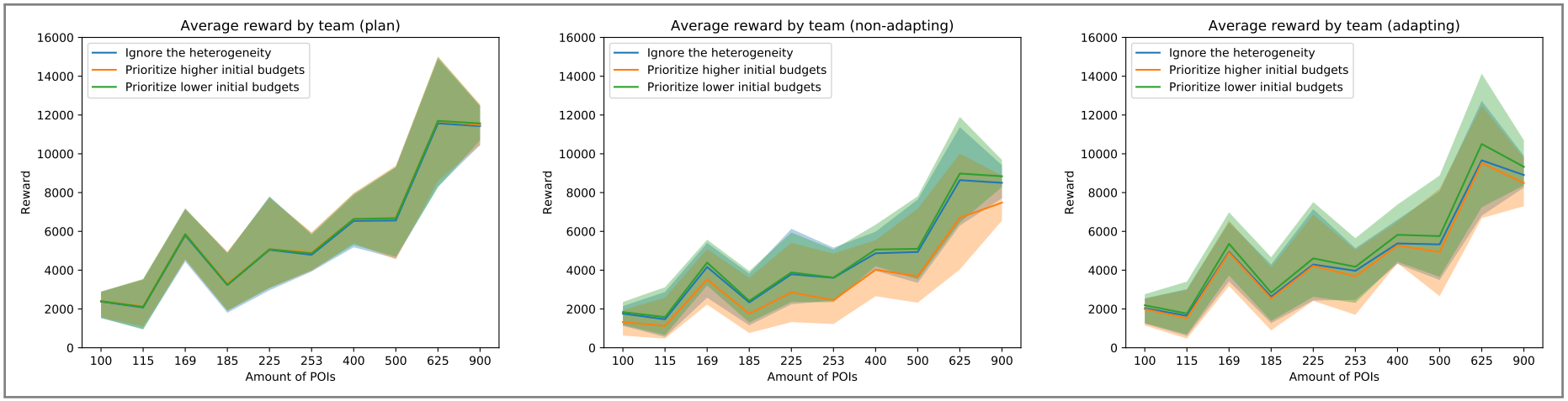
We set up all possible four-member teams using this set of robot types, ensuring that all combinations have at least one SR (WINGS type). We planned and simulated monitoring missions for these teams in environments with more important regions to inspect than others. To verify the best criteria in each process where the heterogeneity

can impact the generated plans with our methodology, we made different implementations and tested them with the built teams. When choosing the agent's selection order to define an initial route between one level and another in region-level planning, we adopt ordering the agents by the initial budget amount. As for the criteria used by the SELECTAGENT function when defining the agent with the highest priority to visit a given POI, we use three strategies:

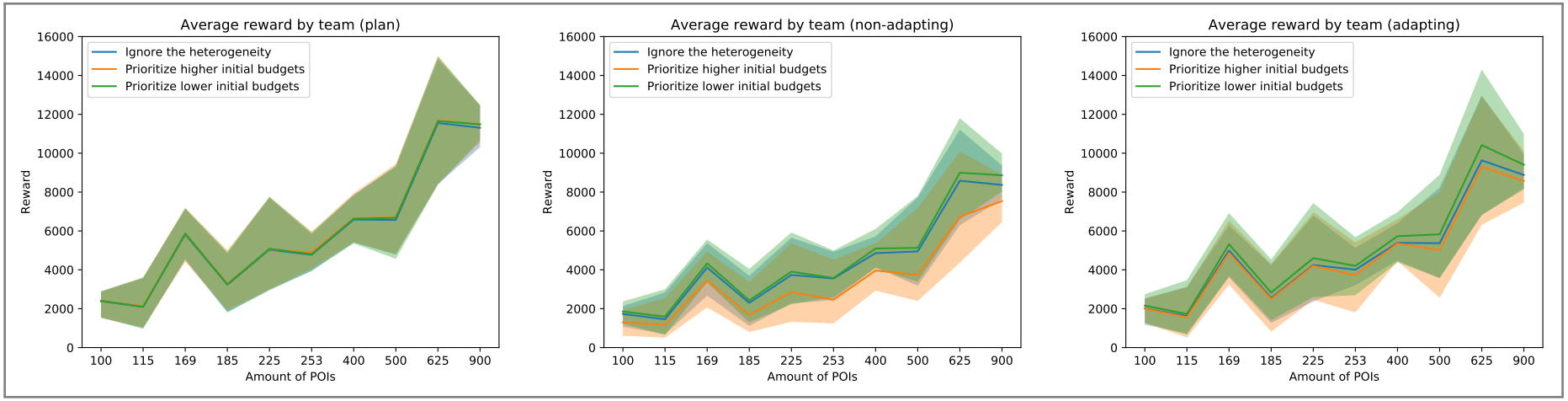
- Ignore the heterogeneity: ignores that agents have different budgets and choose the agent with the highest available budget. This strategy is the same one used in planning for homogeneous teams.
- Prioritize lower initial budgets: selects the agent with the highest available budget among those with the lowest initial budget. That is, the agents with the lowest initial budget have priority over the others.
- Prioritize higher initial budgets: selects the agent with the highest available budget among those with the highest initial budget. That is, the agents with the highest initial budget have priority over the others.

Figure 5.3 shows the results obtained for the planning and simulation considering all the teams, grouped by the amount of POIs. In Figure 5.3a, we sort the agents ascending by the initial budget amount to set the initial route in one level in the region-level planning. In contrast, in Figure 5.3b, we sort the agents descendent by initial budget. The first plot of each line represents the average of rewards planned, according to the number of POIs of the instances, for each criterion used in the SELECTAGENT function. The last two plots in each line show the averages of the final rewards collected by the teams after the missions simulation in the scenario where the agent responsible for the highest valuable route failed right at the beginning of the mission. The middle plot shows the results when the other agents ignored the teammate's failure, and the last one is when the robots still active in the mission adapted their routes using the method proposed in Section 4.2.

We can see in the two plots that show the averages of the planned rewards (the first of each line) that the different criteria used practically do not interfere with the result of the obtained plans. This shows that our method generates consistent plans regarding the reward planned for the team. However, when looking at the plots showing the results of simulations where an agent fails, we notice that these criteria start to make more significant differences. When considering that the active agents ignore the partner's failure (middle plots), we see that when more than one agent is interested in sensing the same POI, it is more beneficial to prioritize the agents that have the lowest initial budget to be the responsible for that POI. The worst strategy is prioritizing agents with the highest initial budget at that point. When considering the scenario where agents in the mission adapt their routes (last plots of each line), we notice that, once again, our



(a)

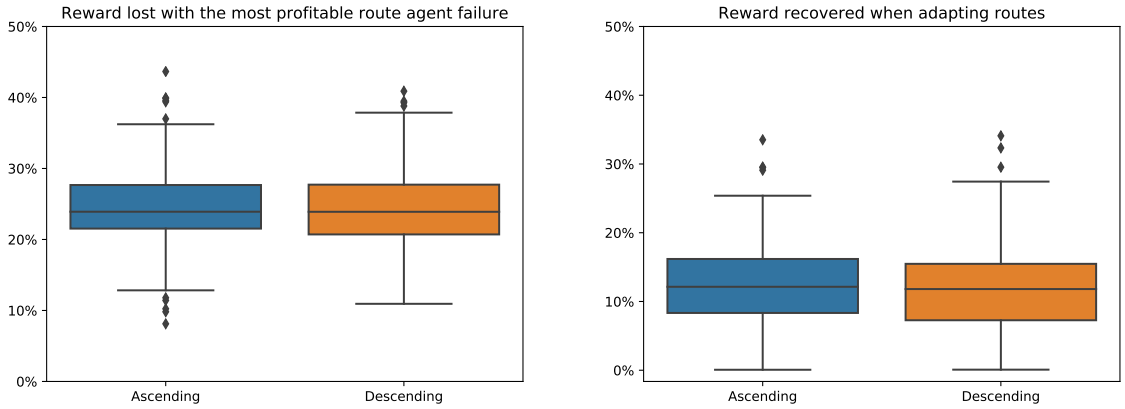


(b)

Figure 5.3: Comparison of computed rewards considering all teams with four members, according to the number of POIs in the instance. Regarding the order of a choice to set the initial route in a level in the region-level planning: (a) agents are sorted in ascending order by starting budget amount; (b) agents are sorted by the initial budget amount in descending order. From left to right, the plotted results represent 1. initial plan; 2. failure ignored by the teammates; 3. The active agents adapt their routes to mitigate the failure.

adaptation routine manages to recover a good part of the reward the team would lose due to the failure. But, regarding the criteria used by the SELECTAGENT function, prioritizing agents with the lowest initial budget is still the best choice.

Regarding the order of choice of agents to set an initial route at a level in the region-level planning stage, we see that the impact is much smaller. From the plots shown in Figure 5.3, deciding the best order to choose the agents is practically impossible. In a more detailed analysis, Figure 5.4 compares the collected reward percentage in failure and route adaptation cases. As mentioned, we consider two strategies in setting the initial routes at a level in the region-level planning phase: ordering the agents by initial budget in ascending or descending order. The ascending strategy first chooses the agents with the lowest initial budgets when setting an initial route. In contrast, the descending approach starts this process with the agents with the highest initial budget.



(a) (b)

Figure 5.4: Percentages of rewards lost with the failure and recovered with the adapting routine. (a) Percentage lost on the robot's failure responsible for the most profitable route; (b) Percentage recovered after failure using the proposed adaptation method.

Figure 5.4a shows the percentage of the planned reward lost when the agent responsible for the most profitable route fails at the beginning of the mission. We noticed that when considering the ascending strategy, the percentage of reward lost with failures varies slightly less than when considering the descending order. Figure 5.4b shows the percentage of planned reward recovered using our route adaptation routine. Once again, considering the ascending order has a slight advantage, that is, prioritizing agents with the lowest initial budget in the planning phase has a slight advantage when it comes to recovering part of the reward that would be lost with the failure of an agent in the mission execution.

Until now, we have discussed results that consider several different team configurations. Overall, our results show that prioritizing agents with smaller initial budgets is the best strategy when we have agents with different budgets. To deepen the analysis, we list some specific teams to highlight some cases:

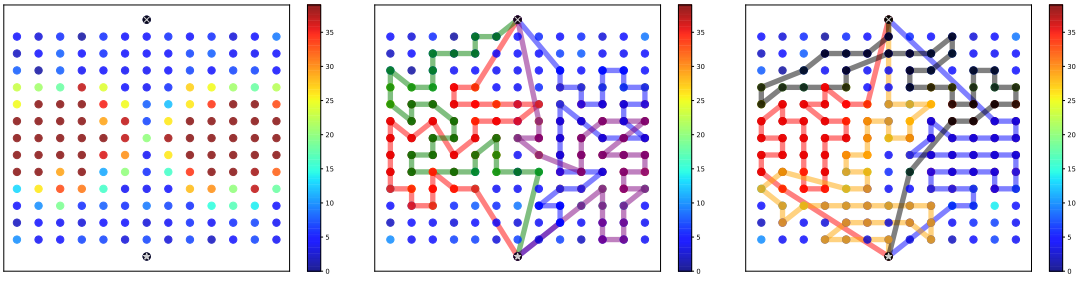
- TEAM-1: team with homogeneous robots. This team allows us to have a comparison parameter against the homogeneous version of the problem discussed in Chapter 4.
  Team composition: WINGS-1 WINGS-2 WINGS-3 WINGS-4

- TEAM-2: team with three equal robots and one robot with 50% more available budget than the others.
  Team composition: WINGS-1 WINGS-2 WINGS-3 TRACKS-1

- TEAM-3: team with two pairs of equal robots. One of the pairs of robots has an available budget 25% higher than the robots in the other pair.
  Team composition: WINGS-1 WINGS-2 LEGS-1 LEGS-2

- TEAM-4: team where all robots have different budgets. Robots have a 25%, 50%, and 100% more available budget than the SR that is also part of the team.
  Team composition: WINGS-1 LEGS-1 TRACKS-1 WHEELS-1

The first analysis was to visualize the routes generated for these teams. Figure 5.5a shows an environment composed of 169 POIs arranged in a grid and with two regions more important to be inspected in the environment central region. The routes plotted in Figure 5.5b were generated for a team where all the robots have the same budget. They have the same characteristics discussed in Chapter 4, and the routes intertwine in the most important regions of the environment to benefit the route adaptation routine in case of any agent failure. In Figure 5.5c, we have the routes generated for the TEAM-3. The red and blue paths are for robots WINGS-1 and WINGS-2, respectively, while the orange and black routes are for robots LEGS-1 and LEGS-2. We see that these routes are slightly less intertwined than the routes of the homogeneous team and that the robots with the lowest initial budget have more responsibility to sense the regions more important to inspect. Robots with higher budgets have routes bordering critical areas and can cover the region in case of a partner's failure. These characteristics were also observed in the routes of the other heterogeneous teams.



(a) Environment (b) TEAM-1 (c) TEAM-3

Figure 5.5: Comparison of routes generated for a homogeneous team and for a heterogeneous team. (a) Environment representation; (b) routes generated for a homogeneous team; (c) routes generated for a heterogeneous team.

To further the understanding of generated routes, we look at the relation between the number of points on the routes versus the reward collected. Figure 5.6 compares the three picked heterogeneous teams. On average, the robots' routes with a higher budget have a greater number of POIs but are less profitable. This is good because the impact is smaller if this agent fails. But suppose the robot responsible for the more promising route fails. In that case, the higher budget robots have a route with many POIs that can be adapted and pass through a valuable region that would be uncovered.

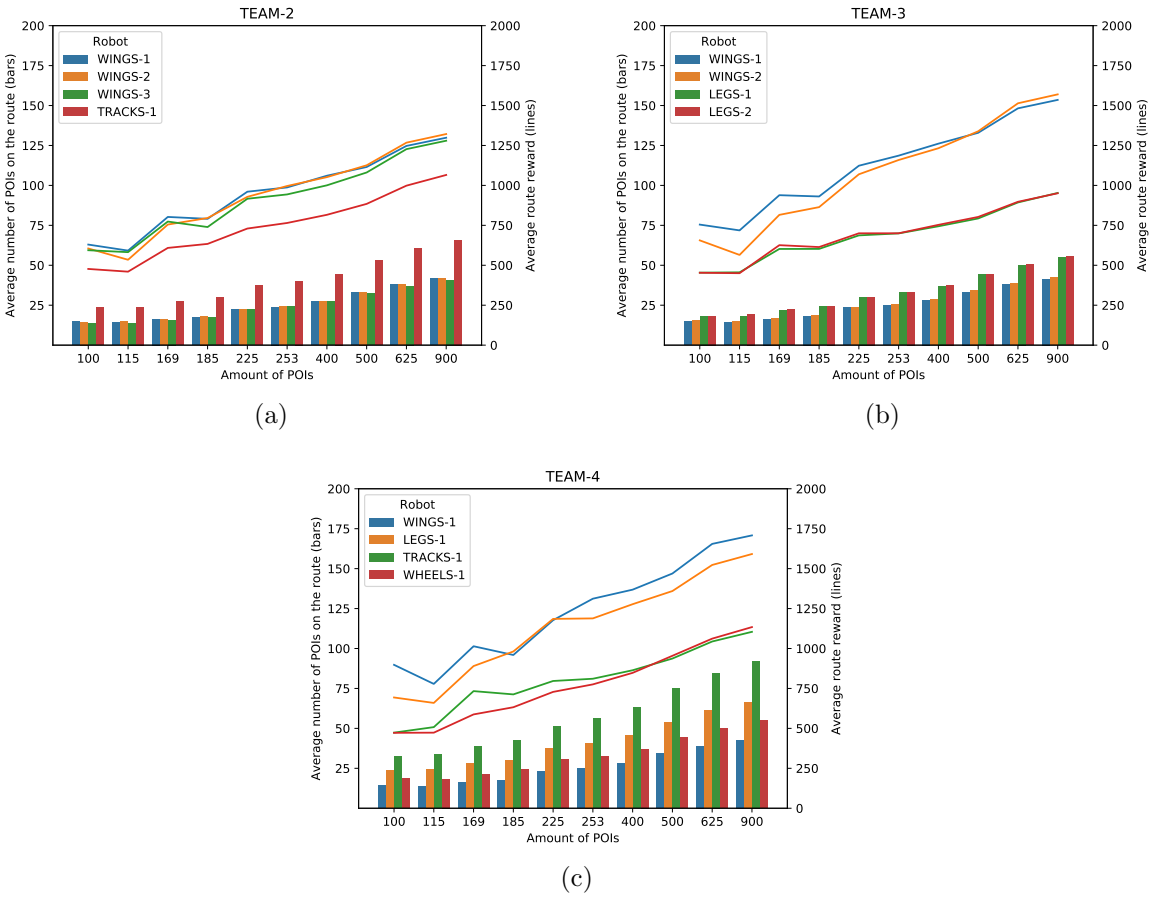


Figure 5.6: Analysis of the routes generated for the picked heterogeneous teams. The average amount of POIs on each route is indicated with the bars (left axis), and the average reward of the routes is represented by the lines (right axis).

We compared these four spotlighted teams in simulations where the robot responsible for the most profitable route in the plan failed right at the beginning of the mission. In the first plot of Figure 5.7, we see the average reward planned according to the instances' number of POIs. As expected, the teams that have members with bigger budgets have a more profitable plan. In the second plot, we have the result when teammates ignore their partner's failure. We notice the considerable drop in rewards caused by the failure, but there isn't a team that has been much more harmed than another just because they have agents with different budgets. This indicates that our planning method balanced the routes well among team members, regardless of the available budget. The last plot

shows the result of adapting active agents' routes in the mission after the partner's failure. We can see that the teams composed of more agents with bigger budgets recover more rewards than the others. This was also expected because if the robots have more budget, they can better adapt their routes to cover possible failures.

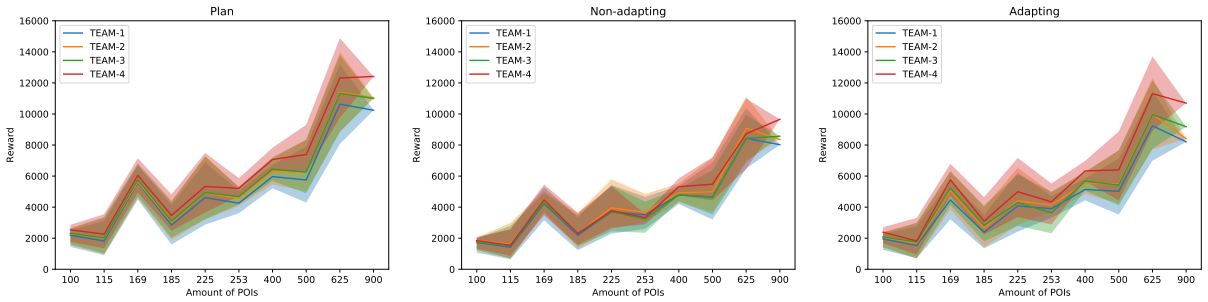
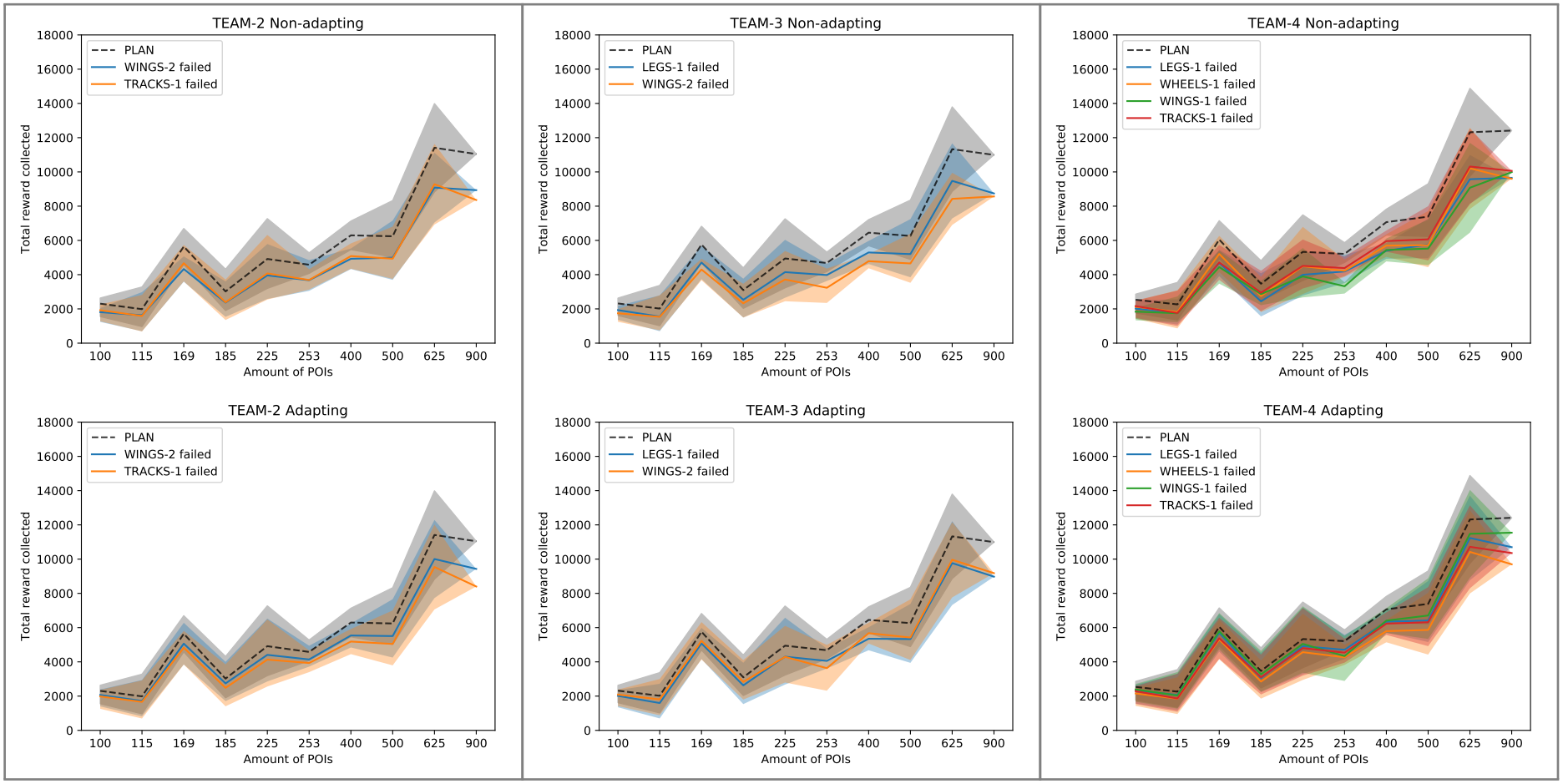


Figure 5.7: Comparison of the computed average rewards considering the four spotlighted teams, according to the number of POIs in the instance.

The results in Figure 5.7 show the failure of the robot that would follow the plan's most profitable route, regardless of the type of robot. In Figure 5.8, we compare when the failure occurs according to the type of robot better to understand the impact of each agent on the mission. In these plots, for each team, we selected among the robots of the same type the one responsible for the most profitable route and simulated its failure at the beginning of the mission. The black dashed lines represent the average planned reward for the team. In the upper part of Figure 5.8a, regardless of the failed agent, the impact on the reward collected was not so different when teammates ignored the failure. By adapting the agents' routes, in the lower plot, we see that when WINGS-2 fails, the presence of TRACKS-1 in the team makes all the difference. As TRACKS-1 has a budget 50% higher, we adapted its route to cover part of the POIs not visited by WINGS-2 and achieved a higher average reward collected than when TRACKS-1 fails.

In Figure 5.8b, we have two pairs of agents with the same budget in the team. When LEGS-1 fails, and the other robots keep their paths, the reward collected at the end is greater than when WINGS-2 fails. This is expected since we prioritize the agents with the lowest budget when planning, so WINGS type robots are responsible for visiting the most valuable POIs. However, adapting the agents' routes after the failure shows similar results, regardless of the failed robot. This is because when the WINGS-2 fails, the two LEGS-type robots on the team, which have 25% more budget, adapt their routes and cover the points not visited by WINGS-2. In Figure 5.8c, we have a team where all robots have different budgets. When agents ignore companion failure, the biggest reward loss is when lower-budget agents fail. However, this scenario is reversed when the robots adapt their routes after the failure. When a robot with a small budget fails, the rest of the team will have enough budget to adjust their routes and recover a good part of the reward that would be lost.



(a) (b) (c)

Figure 5.8: Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance. For each type of agent on the team, we force the failure of the one responsible for the most profitable route. The upper plots show when the team members ignore the partner's failure, and the lower when the agents adapt their routes. The black dashed line corresponds to the planned reward for the team.

This Section showed that considering different budgets when planning routes can reduce the impact of possible failures compared to when this heterogeneity is ignored. Furthermore, in the planning process, we saw that prioritizing agents with the lowest budget when assigning a POI to a route is the best strategy in the general case. If this agent fails, the others will have more budget to adapt their routes, and if it doesn't fail, it will cover the important POIs assigned to its route itself. The result of the plans is lower budget agents with smaller but more profitable routes, while higher budget agents have routes with more POIs but less valuable. We also saw that the choice order of the agents when setting an initial route at a level in the region-level planning phase has a low impact on the method's behavior. We believe this occurs because this process is done only in the first planning phase to direct the agents to the regions of the environment. After this process, many insertions and removals of POIs are still made in the routes, which lessens their influence on the final result of the mission.

5.2.2 Heterogeneous perception

Another heterogeneity analyzed was the agents' perception capability, representing the information quality collected by the sensor and the correlation range of this sensor, that is, the amount of correlated information from neighboring POIs it manages to capture. In this analysis, we seek to understand the impact of having robots with different perception capabilities in the same mission and how our method would behave when generating routes according to the robot's ability to capture information from the environment. Without loss of generality, we assume that all robots have the same maximum speed and starting budget. Thus, the only characteristics differentiating the robots are related to the sensor it is equipped with. Table 5.7 summarizes the types of robots considered in this experiment and the characteristics of their sensors.

We followed the same steps taken in the experiments in Section 5.2.1. For brevity and reader engagement, we emphasize the key findings here and provide a comprehensive description in Appendix A. We planned and simulated monitoring missions for all possible four-member teams using this set of robot types, ensuring that all combinations have at least one SR (QUAD type). It is noteworthy that, in these experiments, although we are considering two characteristics of the sensors, the way we assume them in Table 5.7 makes them inversely proportional. The higher the quality of the information captured, the lower the correlation range and vice versa. Thus, ordering the agents in ascending order considering the information quality, is equivalent to ordering them in descending order by the correlation range. That's why, when choosing the agent's selection order to

Table 5.7: Considered robots in the experiments with heterogeneous perception. The classification was given according to the robot that can carry the sensor in the example.

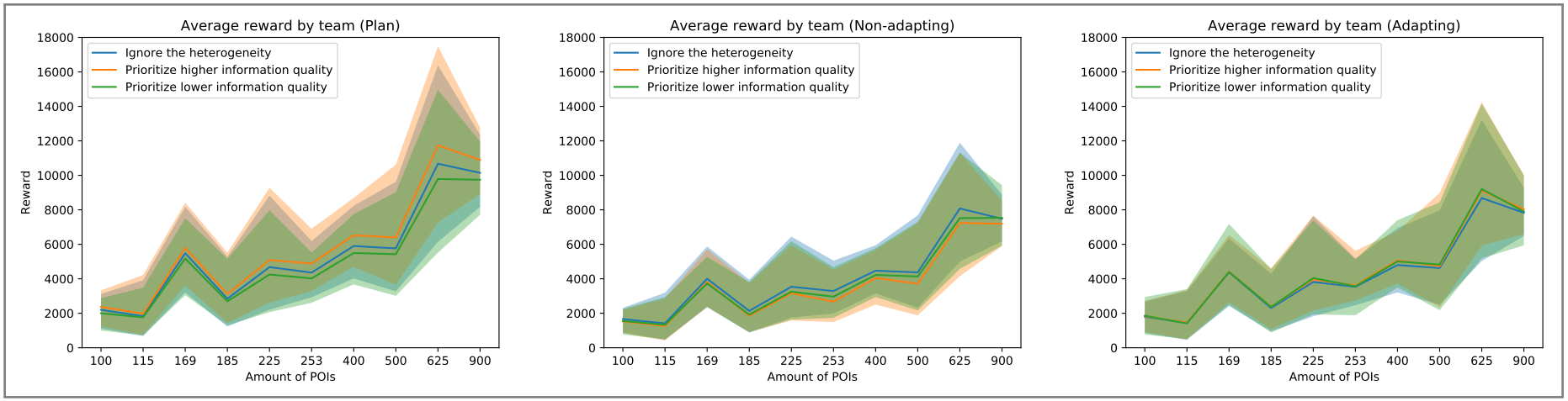
Robot ID	Robot class	Information quality	Correlation range	Sensor example
HWINGS	A-FW	◐○○○	●●●●	RGB camera (at 100m)
LWINGS	A-FW	●○○○	●●●○	Thermal camera (at 60m)
QUAD*	A-MR	●●○○	●●○○	Multispectral camera (at 40m)
TRACKS	G-RT	●●●○	●◐○○	RGB-D camera
LEGS	G-RL	●●●●	●○○○	Soil moisture and temperature

* Standard Robot configuration.

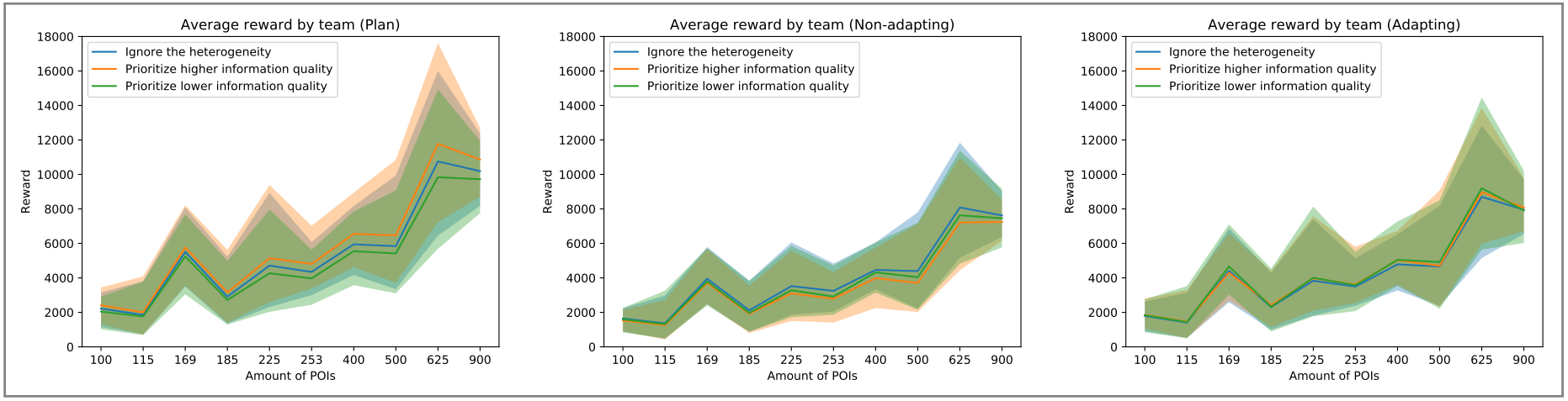
define an initial route between one level and another in region-level planning, we adopt ordering the agents by its sensor information quality. As for the criteria used by the SELECTAGENT function when defining the agent with the highest priority to visit a given POI, we use three strategies:

- Ignore the heterogeneity: ignores that agents have different sensors and choose the agent with the highest available budget. This strategy is the same one used in planning for homogeneous teams.
- Prioritize lower information quality : selects the agent with the highest available budget among those with the lowest sensor information quality. That is, agents with sensors that are not as accurate but cover a greater correlation range have priority over the others.
- Prioritize higher information quality: selects the agent with the highest available budget among those with the highest sensor information quality. That is, agents with very accurate sensors that cover a smaller correlation range have priority over the others.

Figure 5.9 shows the results for the planning and simulation considering all the teams, grouped by the amount of POIs. The two first plots of each line in Figure 5.9 show that depending on the used criteria, we have a greater variation of the planned reward, different from what we observed with the heterogeneous budgets in Section 5.2.1. This variation was expected because we are verifying a criterion defining the agent responsible for a POI with many interested parties, indicating that this POI is more valuable than the others. If we prioritize robots with inferior sensors to inspect this point, the final reward will be smaller than in the case where we prioritize robots with more accurate sensors. However, when observing the plots that show the results of the simulations where an agent fails, we notice that considering the heterogeneity of agents' perception in the planning



(a)



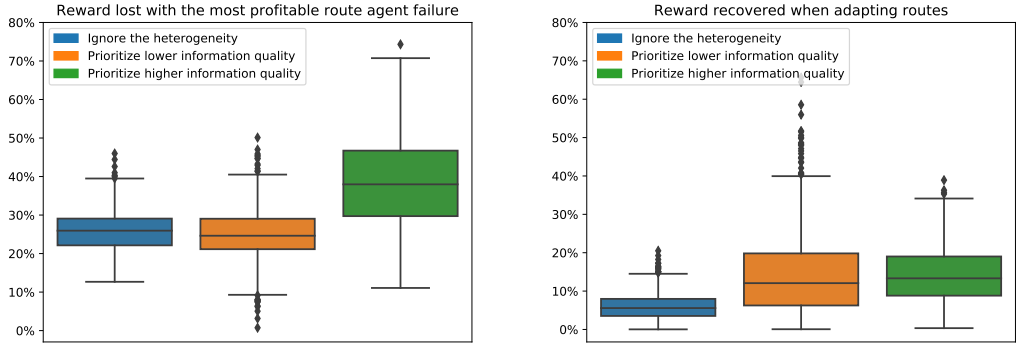
(b)

Figure 5.9: Comparison of computed rewards considering all teams with four members, according to the number of POIs in the instance. Regarding the order of a choice to set the initial route in a level in the region-level planning: (a) agents are sorted in ascending order by sensor information quality; (b) agents are sorted by the sensor quality in descending order. From left to right, the plotted results represent 1. initial plan; 2. failure ignored by the teammates; 3. The active agents adapt their routes to mitigate the failure.

phase no longer generates such expressive gains. In the central plots, the best strategy is to ignore the heterogeneity in the planning phase when the teammates ignore the partner's failure. When we ignore perception heterogeneity in planning, we are only concerned with the budget available for each robot. This strategy intertwines the routes slightly more through the most important regions of the environment, mitigating the lost reward due to the failure of an agent. When considering the scenario where mission agents adapt their routes (the last portions of each line), we realize that the results of the three strategies are even more similar. However, it is possible to notice a small advantage in considering the perception heterogeneity instead of simply ignoring it.

The decision of the best strategy was not as simple as in the case of heterogeneous budgets. Prioritizing agents with the most accurate sensors and lowest correlation range yields the most profitable plans. However, if the robot responsible for the most valuable route fails, planning without considering perception heterogeneities is better. When adapting the routes, we noticed that considering heterogeneity is the best option, but it is unclear in the graph which type of sensor is better to prioritize. Therefore, we have grouped these results in another chart in Figure 5.10. In the context of the problem we are studying, which considers that there will be a failure, and considering the analyzed results, we chose to prioritize robots equipped with sensors with lower information quality. We made this choice because when a failure occurs, the loss of reward is smaller, and we managed to recover a considerable portion of the reward with our adaptation routine.

As an aside, in the middle box of Figure 5.10b, some outliers indicate a reward recovery greater than what was lost according to the central box on Figure 5.10a. It may seem strange, but this really can happen. Just consider that a robot with a better-quality sensor will adapt its route to visit some POIs discovered by a teammate with a worse sensor that failed.



(a) (b)

Figure 5.10: Percentages of rewards lost with the failure and recovered with the adapting routine. (a) Percentage lost on the robot's failure responsible for the most profitable route; (b) Percentage recovered after failure using the proposed adaptation method.

Regarding the order of choice of agents to set an initial route at a level in the

region-level planning stage, we see that the impact is imperceptible. In a closer analysis, detailed in Appendix A, the best strategy to set the initial routes at a level in the region-level planning phase is ordering the agents by sensor information quality in ascending order. That is, it is better to prioritize the agents with the sensors that provide the lowest information quality.

In a qualitative analysis, we saw that the robots with lower quality sensors have more responsibility to sense the regions more important to inspect. Robots with better sensors but with a smaller correlation range have routes bordering critical areas and can cover the region in case of a partner's failure. At first, this strategy may seem counterintuitive, as if robots with higher quality sensors were responsible for the most important regions of the environment, the reward collected at the end would be much greater. However, assuming that a failure will occur, the planned reward loss for the team could be very drastic in these cases. As we aim to reduce the impact of failures and generate routes that benefit the real-time adaptation routine, sending agents with lower-quality sensors to these regions makes more sense. If these agents fail, the teammates will adapt the routes and collect the information that would be lost. If they do not fail, coverage of these regions will be guaranteed, even with a lower quality.

Analyzing the relation between the number of points on the routes versus the reward collected, we do not find a direct relationship. Since all robots have the same budget, the routes have, on average, the same amount of points to be inspected. On average, the routes of robots with better quality sensors yield a lower reward, indicating that those robots visit more POIs outside critical regions. This strategy tries to compensate for the collected rewards and lessen the impact of possible agent failures.

Looking at some specific teams, we highlight that composed by the SR, one LWINGS-1 with a sensor with a quality 25% lower than SR, and robots TRACKS-1 and LEGS-1 with 25% and 50% more quality in your sensors compared to SR, respectively. The presence of LWINGS-1 in the team greatly affects the reward planned since it will be responsible for the most important POIs to be inspected. So, when the teammates ignore their partner's failure, this team performs poorly because it has only one agent responsible for the most important POIs. In contrast, when the route adaptation routine is used after the partner's failure, this team has a big reward recovery because it has agents with better-quality sensors that will adapt their routes. While the outcomes were not discrepant across the various teams, it became apparent that teams with a well-proportioned distribution of agents from each type proved more advantageous. Such balanced formations mitigate the risk of overburdening a single agent within the initial plan, and in the event of a failure, the losses are less severe. Furthermore, this configuration yielded favorable outcomes in terms of route adaptation.

This Section has shown that considering different perception capabilities when planning routes can reduce the impact of potential failures compared to when this hetero-

geneity is ignored. Furthermore, in the planning process, prioritizing agents with sensors that collect information with lower quality and higher correlation range when assigning a POI to a route is the best strategy in the general case. As discussed when analyzing budget heterogeneity in Section 5.2.1, the agents' choice order when defining an initial route at a level in the region-level planning has a low impact on the method's behavior. In fact, the results considering the two heterogeneities, are very similar. Agents with inferior resources are prioritized in the planning and are responsible by the regions with the most important points to be inspected. In case of these agents' failure, the routes' adaptation will be carried out by robots with better resources, and it will be possible to recover a large part of the reward that would be lost. If these agents do not fail, they will guarantee coverage of important POIs. A more detailed version of the analyzes presented here is presented in the Appendix A.

5.2.3 Heterogeneous costs

We also analyzed the heterogeneity of the costs that the agent has to carry out its tasks: the time it takes to move from one point to another and the time spent to collect information in a POI. Although the sensing time is directly linked to the sensor and not to the agent itself, we analyzed it separately from the other sensor characteristics because it is directly related to budget consumption, measured in time. The travel time between two points reflects the agents' speed.

Despite evaluating the speed of agents in this Section, our method does not generate trajectories considering such speeds. Our approach is not concerned with possible collisions between agents at route crossings. We chose to keep only the robot routing without new restrictions to maintain coherence with the other experiments in the mean of not adding more factors that could interfere with results and believing that the achieved results will be good indications for a future method extension considering trajectories.

To better understand the impact of these characteristics on our planning method, we separated these experiments into three groups of heterogeneity: first, we considered speeds, then sensing times, and finally, we considered both heterogeneous costs at the team. For each of these groups, we followed the same steps as the experiments in the previous sections. For brevity and reader engagement, we emphasize the key findings in the following subsections and provide a more detailed description of the analyses in Appendix B.

Different speeds

In this analysis, we tried to understand the impact of having robots with different speeds in the same mission and how our method would behave when generating routes according to the robot's speed to move from one point to another. Without loss of generality, we assume all robots have the same initial budget and are equipped with the same sensor to get information about the environment, that is, have the same sensing time. Thus, the only feature that differentiates robots is their speed. Table 5.8 summarizes all robot types considered in this experiment.

Table 5.8: Considered robots in the experiments with heterogeneous speeds. The classification was given exclusively about maximum speed.

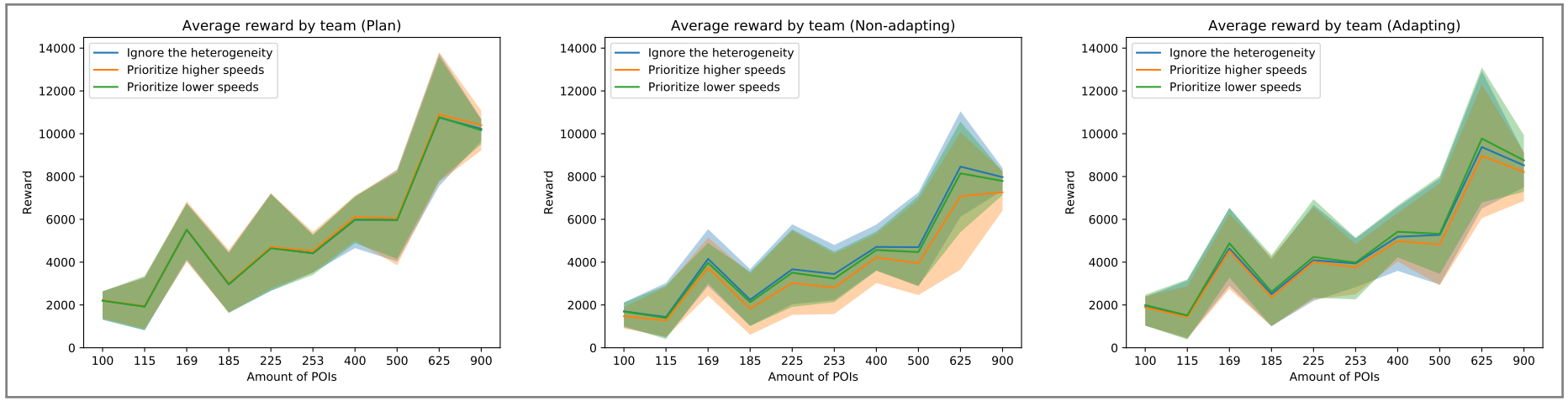
Robot ID	Robot class	Maximum speed	Sensing time
TRACKS	Ground Robot Tracked (G-RT)	●○○○	●●○○
WHEELS*	Ground Robot Wheeled (G-RW)	●●○○	●●○○
QUAD	Aerial Multi-Rotor (A-MR)	●●●○	●●○○
WINGS	Aerial Fixed-Wing (A-FW)	●●●●	●●○○

* Standard Robot configuration.

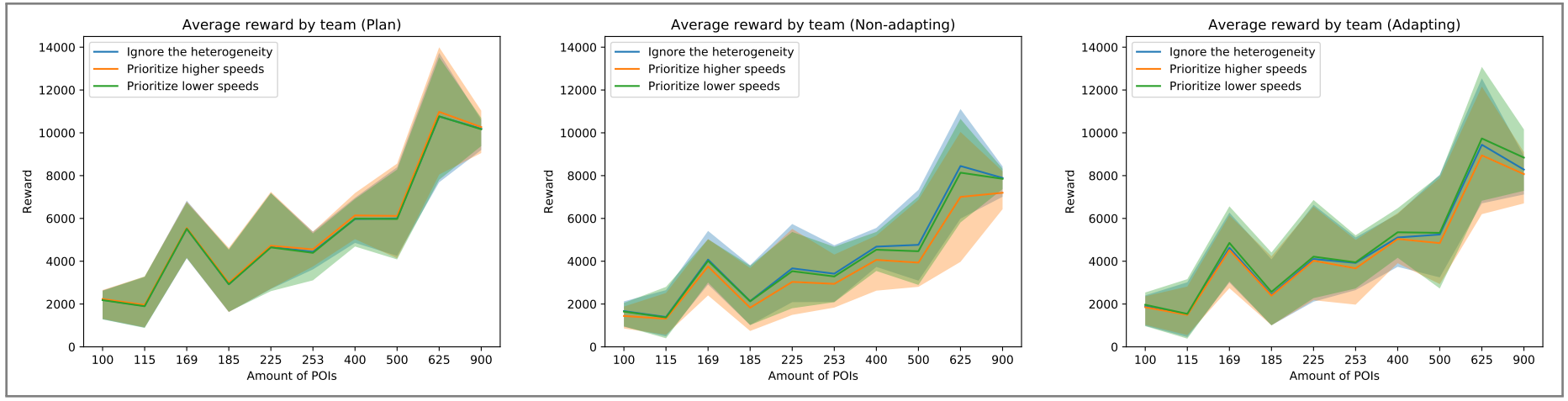
We planned and simulated monitoring missions for all possible four-member teams using the robot types in Table 5.8, ensuring that all combinations have at least one SR (WHEELS type). As criteria used by the selectAgent function when defining the agent with the highest priority to visit a given POI, we use three strategies:

- Ignore the heterogeneity: ignores that agents have different speeds and choose the agent with the highest available budget. This strategy is the same one used in planning for homogeneous teams.
- Prioritize lower speeds: selects the agent with the highest available budget among those with the lowest speed. That is, the slowest agents have priority over the others.
- Prioritize higher speeds: selects the agent with the highest available budget among those with the highest speed. That is, the fastest agents have priority over the others.

Figure 5.11 shows the results obtained for the planning and simulation considering all the teams by the amount of POIs. We can see in the first two plots of each line that the different criteria used practically do not interfere with the result of the obtained plans. However, when looking at the plots showing the results of simulations where an



(a)



(b)

Figure 5.11: Comparison of computed rewards considering all teams with four members, according to the number of POIs in the instance. Regarding the order of a choice to set the initial route in a level in the region-level planning: (a) agents are sorted in ascending order by starting budget amount; (b) agents are sorted by the initial budget amount in descending order. From left to right, the plotted results represent 1. initial plan; 2. failure ignored by the teammates; 3. The active agents adapt their routes to mitigate the failure.

agent fails, we notice that these criteria start to make more significant differences. In the middle plots, when considering that the active agents ignore the partner's failure, the worst strategy is to prioritize the fastest agents at the planning stage. When we ignore the heterogeneity or prioritize the slowest agents, the reward collected is similar. In the last plot of each line, considering the scenario where agents still active in the mission adapt their routes, we notice that, once again, the result of the three strategies does not differ too much, and our adaptation routine manages to recover a good part of the lost reward. But, regarding the criteria used by the SELECTAGENT function, prioritizing the slowest agents is still the best choice. Regarding the order of choice of agents to set an initial route at a level in the region-level planning stage, one more time, we see that the impact is much smaller. We notice that prioritizing the slowest agents when setting an initial route has a slight advantage when it comes to recovering part of the reward that would be lost with the failure of an agent in the mission execution. A detailed analysis is described in Section B.1.

Prioritizing agents with lower speeds, the slower robots have more responsibility to sense the regions more important to inspect, and the fastest ones have routes bordering critical areas and can cover the region in case of a partner's failure. On average, the fastest robots have more POIs in their routes but are less profitable. This was expected because if we have a budget measured in time, a faster robot with the same budget can visit more points than a slower robot. But we noticed that the difference between the number of POIs on the routes is less significant than when we consider agents with different budgets in Section 5.2.1. In addition, there is an interesting situation in Figure 5.12. TRACKS-1 is the slowest robot on the team, with 50% less speed than the SR. This robot should have the most profitable route on the team because it would have priority to inspect the most valuable points in the environment, but that's not what the line chart shows. Priority is given to the TRACKS-1 in the planning, but as it is very slow, the time spent traveling from one point to another makes it unable to inspect many points due to budget constraints. It is possible to see in the bar chart of Figure 5.12 that this agent has the routes with the lowest average number of POIs in all instances; even so, when we have a higher number of POIs, the routes of this agent are more profitable than the fastest robot in the team.

Comparing the teams' performance, the difference in reward lost depends on the failed agent. Ignoring the partner's failure, the reward collected at the end is greater when the fastest robot fails. Since we prioritize the slower agents when planning, lower robots are responsible for visiting the most valuable POIs, and when one of these robots fails, the loss is greater. However, adapting the agents' routes after the failure shows similar results, regardless of the failed robot, as the faster agents manage to visit discovered valuable points.

This Section has shown that considering different speeds when planning routes can

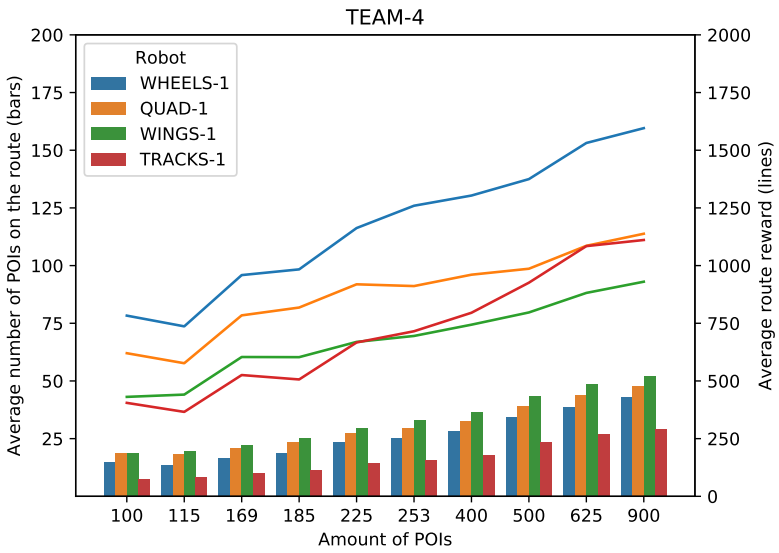


Figure 5.12: Analysis of the routes generated for the TEAM-4. The average amount of POIs on each route is indicated with the bars (left axis), and the average reward of the routes is represented by the lines (right axis).

reduce the impact of potential failures compared to when this heterogeneity is ignored. Furthermore, in the planning process, prioritizing the slowest agents when assigning a POI to a route is the best strategy in the general case. As discussed in the other heterogeneities analyzed, the agents' choice order when defining an initial route at a level in the region-level planning has a low impact on the method's behavior. The results analyzed in this Section are similar to those when considering the other heterogeneities. Agents with inferior resources are prioritized in the planning and are responsible by the regions with the most important points to be inspected. In case of these agents' failure, the routes' adaptation will be carried out by robots with better resources, and it will be possible to recover a large part of the reward that would be lost. If these agents do not fail, they will guarantee coverage of important POIs. A more detailed version of the analyzes presented here is described in the Section B.1.

Different sensing times

In this analysis, we tried to understand the impact of having robots with different sensing times in the same mission and how our method would behave when generating routes according to the robot's spent time capturing environment data. Without loss of generality, we assume all robots have the same initial budget and maximum speed. The information quality and correlation range of the sensor also is the same. Thus, the only feature differentiating robots is the time spent to get data from the environment, the sensing time. Table 5.9 summarizes all robot types considered in this experiment.

We planned and simulated missions for all possible four-member teams using this set of robot types, ensuring that all combinations have at least one SR (QUAD type).

Table 5.9: Considered robots in the experiments with heterogeneous sensing times. The classification was given exclusively about the type of agent that could take that time to sense a point.

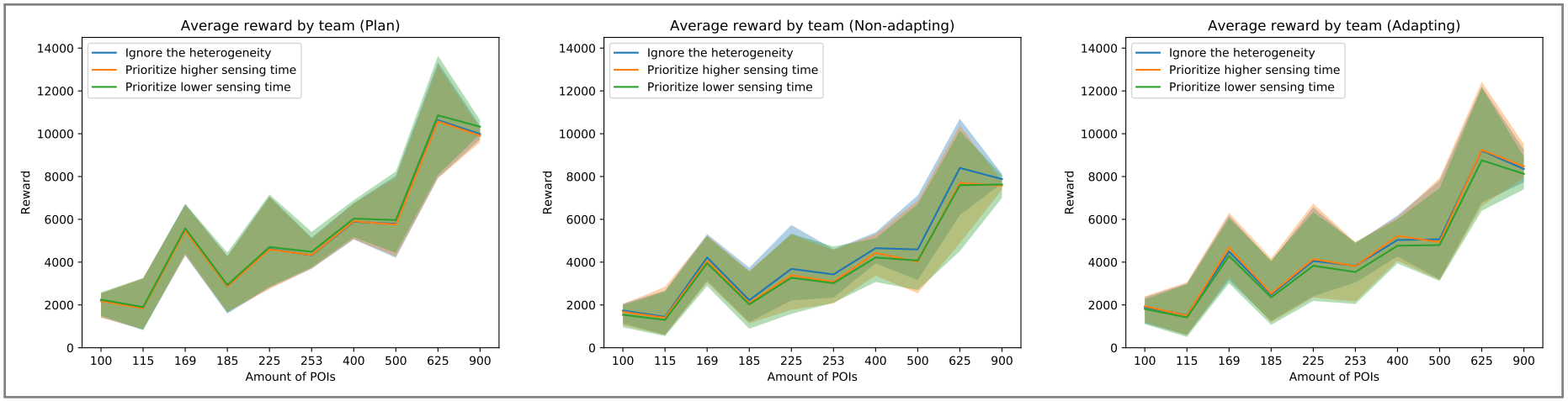
Robot ID	Robot class	Maximum speed	Sensing time
WINGS	Aerial Fixed-Wing (A-FW)	●●○○	●○○○
QUAD*	Aerial Multi-Rotor (A-MR)	●●○○	●●○○
WHEELS	Ground Robot Wheeled (G-RW)	●●○○	●●●○
LEGS	Ground Robot Legged (G-RL)	●●○○	●●●●

* Standard Robot configuration.

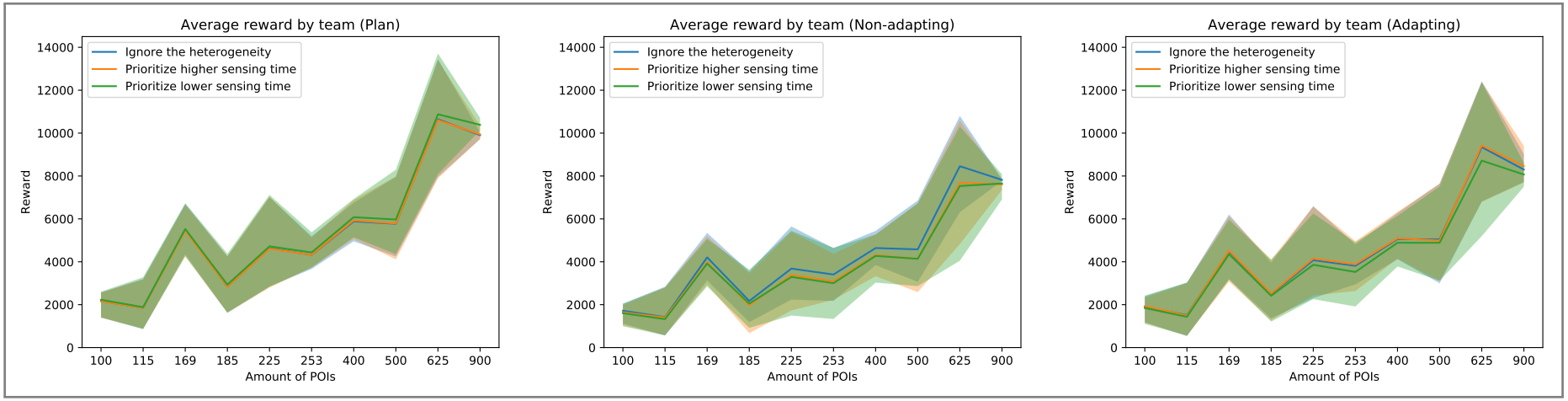
As criteria used by the SELECTAGENT function when defining the agent with the highest priority to visit a given POI, we use three strategies:

- Ignore the heterogeneity: ignores that agents have different sensing times and choose the agent with the highest available budget. This strategy is the same one used in planning for homogeneous teams.
- Prioritize lower sensing times: selects the agent with the highest available budget among those with the lowest sensing time. That is, the agents that take the least time to collect information at a point have priority over the others.
- Prioritize higher sensing times: selects the agent with the highest available budget among those with the highest sensing time. That is, the agents that take the longest time to collect information at a point have priority over the others.

Figure 5.13 shows the results obtained for the planning and simulation considering all the teams by the amount of POIs. We can see in these plots that this heterogeneity the least interferes with the results. The different criteria practically do not impact the obtained plans' results in the first two plots of each line. In the middle plots, when considering that the active agents ignore the partner's failure, the best strategy is to ignore the heterogeneity and only worry about the agents' available budget. Both strategies have slightly inferior performance when considering the agents' sensing time. In the last plot of each line, considering the scenario where agents still active in the mission adapt their routes, the result of the three strategies does not differ too much, and our adaptation routine manages to recover a good part of the lost reward. But, regarding the criteria used by the SELECTAGENT function, at this time, prioritizing agents with the lowest sensing time is the worst choice. Ignoring the heterogeneity and prioritizing higher sensing times take a slight advantage when adapting the routes. In a deeper analysis, we conclude that prioritizing those agents that spend the most time collecting information



(a)



(b)

Figure 5.13: Comparison of computed rewards considering all teams with four members, according to the number of POIs in the instance. Regarding the order of a choice to set the initial route in a level in the region-level planning: (a) agents are sorted in ascending order by starting budget amount; (b) agents are sorted by the initial budget amount in descending order. From left to right, the plotted results represent 1. initial plan; 2. failure ignored by the teammates; 3. The active agents adapt their routes to mitigate the failure.

from the environment is the better choice. Regarding the order of agents choosing to set an initial route at a level in the region-level planning, it is a very difficult decision one more time. Analyzing the results, we can conclude that the descending ordering of agents in the region-level planning is the best strategy for this process. A detailed analysis that endosses these choices is described in Section B.2.

Prioritizing robots with higher sensing costs makes these robots the most responsible for sensing the regions more important to inspect. The robots that take less time to capture information have routes bordering critical areas and can cover the region in case of a partner's failure. On average, the robots that spend more time sensing a POI have routes with fewer points, but they are the most profitable routes in the plan. Indeed, spending more time inspecting a point will result in less available time to move between POIs. But we noticed that the difference between the number of POIs on the routes is less significant than when we consider agents with different speeds in the previous section, which indicates that the sensing time has less impact on planning than the robots' speed.

Comparing the teams' average performance, there is no significant difference between the average rewards planned, regardless of the team for which the plan was made. When agents ignore the teammate's failure, the reward lost grows according to the sensing time of failed robot. That is, the greater the sensing time of the agent that fails, the greater the team's loss of reward. This reflects our planning method, which generates routes prioritizing slower agents to sense. However, this scenario is reversed when the robots adapt their routes after the teammate's failure. The more active agents with shorter sensing time adjusting their routes, the greater the reward the team recovers. Results showed that when robots adapt their routes, the average reward collected is similar regardless of the team, as observed in the case of planned rewards.

This Section has shown that considering different sensing times when planning routes can reduce the impact of potential failures compared to when this heterogeneity is ignored. However, the impact caused by this heterogeneity is smaller than the others analyzed. In the planning process, prioritizing the agents that spend more time to sense a point when assigning a POI to a route is the best strategy in the general case. As discussed in the other heterogeneities analyzed, the agents' choice order when defining an initial route at a level in the region-level planning has a low impact on the method's behavior. Analyzing the case of different sensing times, we can say that an agent that spends less time sensing a point has more time to perform other tasks. In other words, it has more budget. This justifies that this heterogeneity is the first where prioritizing higher values is more beneficial. Higher sensing time here equates to having an agent with a lower budget or velocity. Thus, these results maintain the coherence of what has been seen so far that, when considering heterogeneity, it is best to prioritize agents with lower resources in planning. A more detailed version of the analyzes presented here is described in the Section B.2.

Different speeds and sensing times

We analyzed the impact of different speeds and varying sensing times separately. These two traits are closely tied to the robots' time budget. This experiment aims to vary these two characteristics on the team's robots and verify which causes a greater impact on missions when an agent fails. Without loss of generality, we assume that all robots have the same starting budget and are equipped with the same sensor to get information about the environment. Thus, the only characteristics that differentiate robots are speed and sensing time. Table 5.10 summarizes all types of robots considered in this experiment.

Table 5.10: Considered robots in the experiments with heterogeneous costs. The classification was given exclusively about maximum speed.

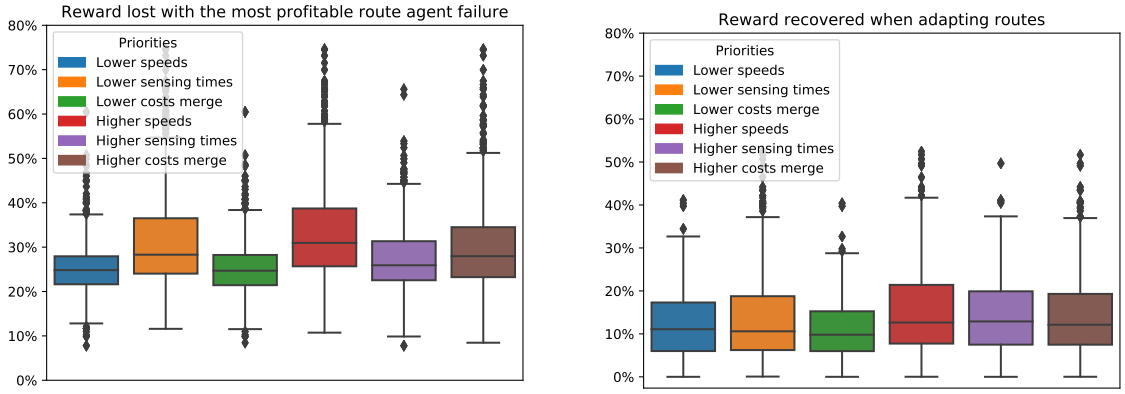
Robot ID	Robot class	Maximum speed	Sensing time
LEGS	Ground Robot Legged (G-RL)	●○○○	●○○○
TRACKS	Ground Robot Tracked (G-RT)	●○○○	●●●●
WHEELS*	Ground Robot Wheeled (G-RW)	●●○○	●●○○
QUAD	Aerial Multi-Rotor (A-MR)	●●●○	●○○○
HEXA	Aerial Multi-Rotor (A-MR)	●●●○	●●●○

* Standard Robot configuration.

We planned and simulated missions for all possible four-member teams using this set of robot types, ensuring that all combinations have at least one SR (WHEELS type). Since we analyze the combination of two characteristics, the number of implementations has increased. We assessed the impact of considering only one of the traits and also tested a strategy that aims to merge both. Increasing the number of implementations made the plots in the same style as the previous sections difficult to comprehend. Therefore, we plotted the results separately to filter the best ones and enable a more readable comparison. Below, we list the considered criteria:

- Velocity: selects agents based only on speed, even if they have other heterogeneities. We tested prioritizing the fastest agents as well as the slowest ones.
- Sensing times: selects agents based only on sensing time, even if they have other heterogeneities. We tested prioritizing agents that take longer and those that take less time to sense a POI.
- Costs merge: selects agents based on a speed and sensing time merge. Our merge function adds the two values representing the cost unit to perform the agent's tasks. We did not delve into studying other functions as it would entail numerous additional experiments, and we believe this function suffices for this work.

Figure 5.14 compares the results obtained with the planning and simulation of missions using all these criteria. In Figure 5.14a, we see the reward percentage lost when the agent responsible for the most profitable route fails right at the beginning of the mission and the rest of the team ignores this event. The results are consistent with what was discussed in previous sections. Figure 5.14b shows the reward percentage recovered when teammates adapt their routes. We must be more careful in this analysis because recovering more rewards is not always the best indicator. For example, the results obtained when prioritizing robots with higher speeds present a higher reward recovery. However, the reward loss is also very high when using this criterion.



(a) (b)

Figure 5.14: Percentages of rewards lost with the failure and recovered with the adapting routine. (a) Percentage lost on the robot's failure responsible for the most profitable route; (b) Percentage recovered after failure using the proposed adaptation method.

Based on the results of Figure 5.14 and in the discussions in our analyses, we pick the best implementation of each criterion to perform a more detailed comparison. That is, we use four strategies in SELECTAGENT function when defining the agent with the highest priority to visit a given POI:

- Ignore the heterogeneity: ignores that agents have different costs and choose the agent with the highest available budget. This strategy is the same one used in planning for homogeneous teams.
- Prioritize lower speeds: selects the agent with the highest available budget among those with the lowest speed. That is, it prioritizes the slowest agents over the others.
- Prioritize higher sensing times: selects the agent with the highest available budget among those with the highest sensing time. That is, the agents that take the longest time to collect information at a point have priority over the others.
- Prioritize lower costs merge: selects the agent with the highest available budget among those with the lower costs merge. That is, the agents with the lower sum of speed and sensing time have priority over the others.

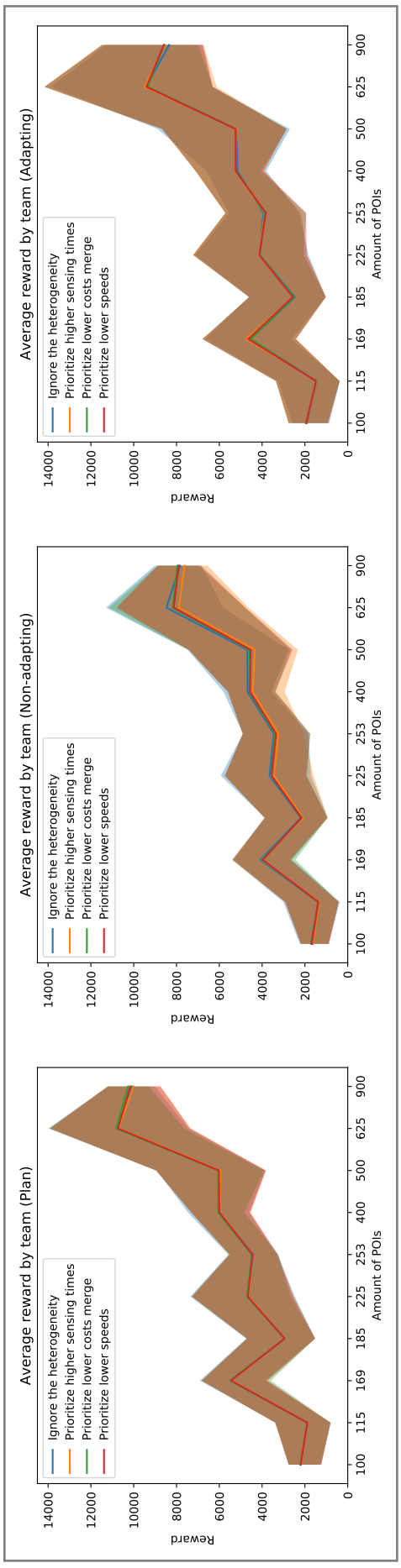
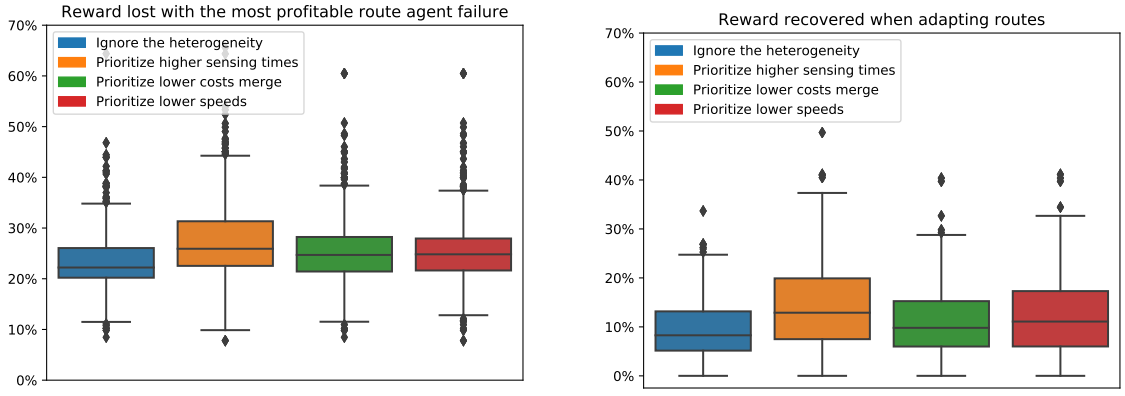


Figure 5.15: Comparison of computed rewards considering all teams with four members, according to the number of POIs in the instance. From left to right, the plotted results represent 1. initial plan; 2. failure ignored by the teammates; 3. The active agents adapt their routes to mitigate the failure.

Figure 5.15 shows the results obtained for the planning and simulation considering all the teams by the amount of POIs. We can see that the different criteria used do not significantly impact the amount of planned reward, which shows that the plans' results are consistent. When considering that the active agents ignore the partner's failure, we see that the differences exist but are not significant. With little advantage, we can note that ignoring the heterogeneity in the planning is the best strategy, while the worst is prioritizing the agents considering only their sensing times. When the agents still active in the mission adapt their routes, we see again that it is almost imperceptible which is the best strategy to be adopted in the planning.

To better understand the results, Figure 5.16 shows these results in a different chart. Figure 5.16a reaffirms the observations from Figure 5.15 plots. Meanwhile, Figure 5.16b demonstrates that prioritizing agents with longer sensing times is the strategy that most retrieves information using our adaptation routine. However, this strategy is also associated with the highest loss of reward in the event of failure. Analyzing both plots in Figure 5.16, we can conclude that prioritizing agents with lower speeds is the best planning strategy. That is, the speed of an agent is a more impactful factor than the sensing time. Regarding the order of choice of agents to set an initial route at a level in



(a) (b)

Figure 5.16: Percentages of rewards lost with the failure and recovered with the adapting routine. (a) Percentage lost on the robot's failure responsible for the most profitable route; (b) Percentage recovered after failure using the proposed adaptation method.

the region-level planning stage, we have also increased the number of possibilities. As we noted the low impact of this criterion in our previous analyses, we disregarded this factor in the results discussed so far to make the charts more readable. More once the difference between the ordering criteria at this stage is imperceptible. Analyzing the averages of loss and recovery, we concluded that the best ordering criterion is ascending by sensing time. A detailed analysis is described in Section B.3.

Considering the best combination of strategies, the routes generated are, on average, more profitable for the slowest robots, even with fewer points. This characteristic

had already been observed when we analyzed the speed separately. But here, we also have the impact of different sensing times. This factor makes the difference in the number of points on the routes less significant than when we analyze only the speed. The teams composed of two SR and with a pair of faster agents were the ones that obtained the highest planned rewards, while the teams with robots slower than SR had the lowest rewards. The sensing time also influenced the planned reward, causing teams with agents that have shorter sensing times to collect more rewards than the others. When teammates ignore the failure, the reward loss is similar for all teams. Using our route adaptation routine, agents of type QUAD calls the most attention because they are the fastest and have the shortest sensing times. For example, the HEXA-type agents are as fast as the QUAD-type agents but take longer to sense a point. This delay reflects on reward recovery. Considering different speeds and sensing times is the first case where considering heterogeneities in planning is not so beneficial when agents adapt their routes at the end of the mission. However, this happens when the team has very slow robots or high sensing times.

In cases where the pair of partners of the two SR are 50% faster than them, the reward lost with the slower agents' failure is greater, as they are responsible for the most profitable routes. As we have seen, having better-resource agents favors the adaptation routine. Robots that are fasters and spend less time sensing the points are particularly advantageous in adapting their routes. When the pair of partners of the two SR are 50% slower than them, the impact caused by the failure of any agent is similar. The sensing times also make a difference in the mission results when we use the route adaptation routine. We observed that the reward recovery by the adaptation routine was not too advantageous for slow robots that spent a long time to sense a point. In contrast, if the robot is fast in sensing tasks, even though it is slow to move, the adaptation routine results in more consistent results at the end of the mission.

This Section showed that considering different speeds and sensing times when planning can reduce the impact of possible failures compared to when these heterogeneities are ignored depending on the team configuration. Teams with agents that are very slow about others generate less profitable plans, and in case of failure of a faster agent, the rest of the team cannot recover much of the lost reward. The results show that both speed and sensing times impact the collected reward. Slower agents and those with higher sensing times cannot adapt their routes to cover possible teammate failures. This reinforces that the best strategy for our planning method is to prioritize agents with fewer resources to be responsible for the regions with the most important points to be inspected. In case of these agents' failure, the routes' adaptation will be made by robots with better resources, making it possible to recover a large part of the reward that would be lost. If these agents don't fail, they will ensure coverage of important POIs. A more detailed version of the analyses presented here is described in Section B.3.

5.3 Concluding remarks

In this chapter, we analyzed how teams composed of agents with different characteristics would behave in monitoring areas with the most important regions to be inspected when one of the team members fails. As far as we know, no work deals with this subject in the literature. Our analysis considered autonomous agents with different uptime autonomy, movement and sensing costs, and perception capability. These characteristics interfere directly with the modeling of our problem. The uptime autonomy concerns the available budget for each agent, that is, the time available to carry out its route. The movement cost is associated with the speed with which the agent moves in the environment, while the sensing cost is equivalent to the time it takes to collect information in the POI. Finally, the agent's perception capability is related to the sensors it has and the quality of these sensors.

Once our proposed methodology for planning and adapting routes considering homogeneous teams had good results, we extended the same idea to heterogeneous teams treating the different characteristics separately. To accomplish this, we maintained the subdivision of the environment into regions and created an RSL for each team member estimating the costs and rewards according to their characteristics. And we also kept the two-phase planning approach.

In the region-level planning, we identified a point that could impact the result when agents have different characteristics. Specifically, when we set an initial route for the agent at a RSL level, using the route calculated at the previous level. We saw with the results that this process has very little impact on the mission's outcome, regardless of the heterogeneity considered. This is because it exists only in the first planning phase, and after that, many insertions and removals will still be carried out.

Another critical point that we identified in our method that could be greatly influenced by heterogeneity and affect the results of the plans was the removal of multiple visits to the same POI. We allow multiple visits to the same point in some moments of our planning methodology, so the removal step is important since only a single agent can visit a POI. Considering heterogeneity when deciding which agent will be responsible for inspecting a POI makes all the difference when we adjust the robots' routes after the failure of a team member.

Since our planning method for homogeneous teams proved effective in cases of failure, we used the plans generated by it as a comparison parameter for those generated for heterogeneous teams. We saw that, except when considering the perception capability, the generated plans are very similar regarding the planned reward for the team. What differentiates the plans is how the routes are spread in the environment. When we consider different perception capabilities, it is evident that teams with better sensors will collect

more rewards than others.

Notably, we propose using the planning method in conjunction with the route adaptation routine we propose. Both together bring the best results for the cases we analyze. However, if the intention is not to use the adaptation routine, the results showed that ignoring the heterogeneities in the planning phase is the best option. This is because, when we do not consider heterogeneity, the routes become more intertwined, as shown in the results of Chapter 4. Considering heterogeneity, our method can lose a lot of rewards depending on the agent that will fail but manages to recover much more using our adaptation routine.

The results showed that the best planning strategy is prioritizing agents with the worst resources to visit the regions most important to inspect in the environment. That is, it is better to prioritize agents that have less uptime autonomy when considering different budgets. When considering distinct perception capabilities, it is better to favor those with sensors that collect lower-quality information. Regarding costs, prioritizing slower agents in terms of speed and those who spend more time gathering information in treating sensing time are the better choices. The premise is that if one of these agents fails, the team will still have members with superior resources who can adapt their routes to cover the points in the failed agent's route. In the event these agent doesn't fail, the team ensures coverage of the regions more important to inspect in the environment.

Regarding the formation of teams, we observed that in those with a similar number of agents of each type, the reward collected at the end of the mission after adapting the routes is similar regardless of the type of agent that fails. This is because agents with better resources manage to adjust their routes very well. The best formation is the one with all the different agents. If an agent with better resources fails, the impact is smaller, and there will still be robots with good resources to adapt the routes. In cases where an agent with fewer resources fails, the loss of reward is greater, but we have more agents with better resources to adapt the routes.

With all that was exposed, it can be concluded that the combination of the proposed planning method and our adaptation routine are great options for heterogeneous teams of agents prone to failure in the execution phase. This study can be a foundation for future methodologies addressing this type of team in area monitoring missions where an agent can fail.

Chapter 6

Conclusions and Future Works

6.1 Conclusions

Area monitoring is an essential task in diverse and important real-world activities such as surveillance and tracking, search and rescue missions, and environmental and agricultural monitoring. In this dissertation, we were interested in monitoring areas with different importance to inspect using a team of heterogeneous agents. We modeled the problem as an instance of the Correlated Team Orienteering Problem (CTOP), which consists of finding a path for each team member in the environment to maximize the collection of rewards without violating the budget constraints of each agent. The CTOP generates routes for team members, considering that all agents will follow what was planned. However, when we operate in the real world, unforeseen events may occur that prevent an agent from going as planned. For example, an agent may encounter sensor failures, movement problems, and environmental blockages, among other obstacles not part of the initial plan.

To deal with this issue, we addressed the CTOP in two phases: planning and executing the mission. In the first phase, we use approaches from the literature to the CTOP that do not consider that failures may occur. Our first effort was in the mission execution phase, where the agents could fail and not comply with the plan. We propose a method to adapt the routes of the active agents in the mission so that important regions that would be neglected due to the failure of a teammate could be covered. Our method has the advantage of running concurrently with the execution of the mission without the need to interrupt it. We can also see our approach as an anytime algorithm since, at any time, we keep the best solution found, and as our method is being executed, we are improving solutions.

We performed several experiments to verify the proposed method's effectiveness in adapting agents' routes. Solving the planning phase using the literature method, we simulated the execution of the mission where the agents went iteratively, following point by point of their routes. At a certain point in the simulation, we excluded an agent from

the mission to reproduce a failure that prevents it from sensing the environment. This failure represents the worst-case of unexpected agent behaviors. At that time, part of the POIs would no longer be visited, compromising the final collection of the team's reward. To compare the efficiency of our method in all tests, we considered three strategies adopted after the failure of one agent:

1. Non-adapt: the remaining agents ignore the teammate's failure and follow the routes assigned to them at the beginning of the mission.

2. Re-planning: we stopped the simulation and reran the heuristic used in the planning phase considering the current state of the environment.

3. Adapt: we applied our method successively to adapt the agents' routes who are still active in the mission.

The results showed that ignoring an agent's failure is the worst possible strategy concerning the final reward collection. They also indicate that, in this regard, the re-planning of the routes achieves, in most cases, a greater collective reward at the end of the mission. Our method recovered a good part of the reward that would be lost by the failure of an agent, significantly exceeding the strategy of ignoring it. Compared to re-planning the routes, our method was slightly less useful at the end of the mission but expended almost half the time without pausing the mission's progress. In other words, our method proved to be the best choice for an online application in terms of cost-benefit. An example of this is that we applied our method without problems in an instance with 900 POIs and a team with ten agents, where up to nine of them failed during the execution of the mission. The re-planning strategy proved impractical, especially when re-planning (failures) increased.

The CTOP objective is to generate routes that maximize the reward collection at the end of the mission. Analyzing the heuristic results obtained in the planning phase, we observed that this objective was achieved, but the routes were unbalanced regarding reward. That is, the solution contained routes that were much more profitable than others. When we consider that an agent can fail, the routes should be balanced so that the team's final reward is not greatly affected in case of any agent's failure. We also observed that our adaptation routine depends on the arrangement of routes in the environment to achieve better results. Regions covered by only one route away from others have more difficulty in being reached. Based on these observations, we propose a route planning method to mitigate the damage caused by agent failures and benefit the route adaptation routine.

Our planning method defines various levels of representation of the environment and involves region-level and POI-level planning. In region-level planning, we allow more than one agent to inspect the same point because it is interesting to have more than one responsible for important regions in case of failures. However, since multiple visits to

the same point are not allowed in CTOP, for each POI visited by more than one agent, we preserve it in the route of the agent with the highest available budget and remove it from the other routes. This choice aims to keep more balanced routes and free up the budget for the next planning stage. In POI-level planning, we start expanding the routes generated in the previous step. In this process, the agents ignore their partners and maximize their reward collections as if they were isolated in the environment. This step can lead to multiple visits to the same point, which we've solved again by keeping the point in the agent's route with the highest available budget. Then the routes go through a process of removing points that are too expensive to be inspected. Finally, for the last time, a route expansion process was made, now considering all agents in the team and restricting multiple visits to the same POI.

Since we do not focus exclusively on maximizing the planned reward but that collected at the end of the mission execution when there are agent failures, the plans generated by our method estimate a reward slightly small than the planned one using the methods of literature. Analyzing the reward collected at the end of the mission execution, our planning approach proved quite effective compared to the other methods. We can say that using our planning method is a great advantage even if team members ignore a partner's failure. Together with the proposed route adaptation method, the final collected reward is still slightly lower than when using other methods. But, we have greatly improved the time taken to generate the plans. These facts make our planning strategy a great choice for planning missions that failure-prone agents will execute.

We also explore the problem of monitoring areas with different importance using teams with heterogeneous agents. We analyze three heterogeneous elements: uptime autonomy, movement and sensing costs, and perception capability. These characteristics interfere directly with the modeling of our problem. We maintained the same planning strategy but made the necessary adjustments to deal with heterogeneous agents. Two processes in our method were identified that are directly influenced by agents' different characteristics. In region-level planning, when we set an initial route for the agent at an RSL level, and when we remove multiple visits to the same POI, which occurs in both planning stages. If we do not consider heterogeneities, we always keep the POI on the agent's route with the highest available budget. However, if we were interested in analyzing another characteristic, the available budget would probably not be a decisive factor in this decision. We modified the agent selection criteria in both processes according to the evaluated heterogeneity.

Results show that considering the heterogeneity in setting an initial route for the agent at an RSL level has very little impact on the mission execution end. This is because it acts only in the first planning phase, and after that, many insertions and removals will still be made. In removing multiple visits to the same POI, we see a lot of difference in the criteria when an agent fails. As we generate routes considering the heterogeneities

and that the proposed adaptation routine will be used, in cases where the agents ignore the partner's failure, the plan assuming the heterogeneity has a small disadvantage. This is because, when we do not consider heterogeneity, the routes become more intertwined in the regions of greater importance, reducing the impact of the failure. However, when using the adaptation routine after the failure, it is possible to recover much more reward than ignoring the heterogeneity.

The analyses indicated that the best strategy is prioritizing agents with the worst resources to inspect the regions more important in the environment. That is, it is better to prioritize agents that have less uptime autonomy when considering the budget. In these cases, agents with small budgets will have routes with fewer POIs but more profitable than those with higher budgets. Regarding the perception capability, the number of POIs on the routes is more balanced. However, agents with sensors that collect lower-quality information have much more profitable routes because they are responsible for the most valuable points in the environment. Related to costs, it is better to favor slower agents and those who spend more time collecting information. The size of the route is also influenced here, but the difference in the number of points in each route is smaller than what happens when we consider different budgets.

Prioritizing the lower resources agents is the best strategy because if one of these agents fails, the team will still have better resource members that will adapt their routes to cover the points that would be uninspected due to the agent's failure. In the event these agent does not fail, the team ensures coverage of the regions more important to be inspected. Concerning team formation, in those with a balanced number of agents of each type, the reward collected at the end of the mission after adapting the routes is similar, regardless of the agent that fails. This is because agents with better resources manage to adjust their routes very well when worst resources robots fail. The best formation is the one with all agents different. If an agent with better resources fails, the impact is smaller, and there will still be robots with good resources to adapt the routes. When an agent with fewer resources fails, we have more agents with better resources to adapt the routes and recover lost rewards.

With all that has been exposed, we conclude that the proposed adaptation routine is an excellent alternative to be used in scenarios where agent failures may occur during mission execution, regardless of the method used to generate the plan. The proposed planning method generated routes that estimate a slightly small reward than the methods in the literature. But, in the event of failure of an agent, the lost reward is much smaller, and the adaptation routine is benefited. The study of the heterogeneity of teams in failure situations was an advance in proposing methodologies that deal with agents' failure in the area monitoring missions.

6.2 Future works

As the treatment of failures related to CTOP is still underexplored in the literature, this work presents several potential future directions that can be pursued. A direct extension of our work would be to consider all the heterogeneities together in the team. We conducted this study by analyzing them separately to understand the impact of each one better. However, considering all the heterogeneities would be a situation that would closely resemble the real world, as the agents are mostly different in all aspects.

Another direct extension of our work would be considering trajectories instead of only routing agents through the environment. As we have observed, our methodologies cross the agents' routes in the environment to mitigate the impact of fails, which can lead to collisions if the agent's position at a given time of mission execution is not considered. Additionally, we have seen that the point at which the agent fails in the mission affects the proposed adaptation method. When considering agents with different speeds and sensing times within the team, considering trajectories would be even more crucial for advancing this topic.

Some considerations that could also be included in the problem modeling would be the possibility of altitude changes in aerial agents and considering the curvature constraints of all agents. If an aerial agent can fly at different altitudes, it may be able to collect information with varying qualities and correlation ranges at the cost of increasing or decreasing the flight altitude. In summary, flights closer to the ground capture better information but with a narrower scope of what is around, as we discussed when addressing different perception capacities. The curvature constraint is also an important characteristic to consider because, for agents with a larger curvature radius, it may not be as feasible to adapt the route to cover nearby points as it would be for agents with a smaller radius.

Furthermore, it would be interesting to consider failures that reduce the robot's capabilities but do not necessarily prevent it from continuing the mission. This is the case, for example, with route deviations to overcome unforeseen environmental obstacles, excessive budget expenditure, or failures in a sensor that reduce the agent's perception capacity. In some of these cases, the robot would not be able to inspect all the points on its route due to budget constraints on tasks that were not planned. Or, in a sensor failure case, it may be able to pass through all the points but collect a lower reward. In all cases, the agents would not abandon the mission and would also need to have their routes adapted to mitigate the impact of the encountered issues.

Bibliography

[1] José J. Acevedo, Begoña C. Arrue, Iván Maza, and Anibal Ollero. Distributed Cooperation of Multiple UAVs for Area Monitoring Missions, pages 471–494. Springer International Publishing, Cham, 2015.

[2] José Joaquín Acevedo, Ivan Maza, Anibal Ollero, and Begoña C. Arrue. An efficient distributed area division method for cooperative monitoring applications with multiple uavs. Sensors, 20(12), 2020.

[3] Amjaad Alhaqbani, Heba Kurdi, and Kamal Youcef-Toumi. Fish-inspired task allocation algorithm for multiple unmanned aerial vehicles in search and rescue missions. Remote Sensing, 13(1), 2021.

[4] E. Angelelli, C. Archetti, and M. Vindigni. The Clustered Orienteering Problem. European Journal of Operational Research, 238(2):404–414, 10 2014.

[5] Enrico Angelelli, Claudia Archetti, Carlo Filippi, and Michele Vindigni. The probabilistic orienteering problem. Computers & Operations Research, 81:269–281, 5 2017.

[6] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics). Princeton University Press, Princeton, NJ, USA, 2007.

[7] Sankalp Arora and Sebastian Scherer. Rapidly exploring Random Orienteering, 2016.

[8] L. Assunção and Geraldo Robson Mateus. A cutting-plane algorithm for the Steiner team orienteering problem. Computers and Industrial Engineering, 135:922–939, 9 2019.

[9] Lucas Assunção and Geraldo Robson Mateus. Coupling Feasibility Pump and Large Neighborhood Search to solve the Steiner team orienteering problem. Computers and Operations Research, 128:105175, 4 2021.

[10] Emilio Barocio, Josue Romero, Ramon Betancourt, Petr Korba, and Felix Rafael Segundo Sevilla. Wide-Area Monitoring of Large Power Systems Based on Simultaneous Processing of Spatio-Temporal Data, pages 189–228. Springer International Publishing, Cham, 2021.

[11] Ana Isabel Barros and Lanah Evers. The Stochastic Scounting Problem. In Proceedings of the Congreso Latino-Iberoamericano de Investigacion Operative-Simposio Brasilairo de Pesquisa Operacional CLAIO-SBPO, 2012.

[12] Behzad Bayat, Naveena Crasta, Alessandro Crespi, António M. Pascoal, and Auke Ijspeert. Environmental monitoring using autonomous vehicles: a survey of recent searching techniques. Current Opinion in Biotechnology, 45:76–84, 2017. Energy biotechnology • Environmental biotechnology.

[13] Zheyong Bian and Xiang Liu. A real-time adjustment strategy for the operational level stochastic orienteering problem: A simulation-aided optimization approach. Transportation Research Part E: Logistics and Transportation Review, 115:246–266, 7 2018.

[14] Nicola Bianchessi, Renata Mansini, and M. Grazia Speranza. A branch-and-cut algorithm for the Team Orienteering Problem. International Transactions in Operational Research, 25(2):627–635, 3 2018.

[15] N. Brasch. Robots of the Future. Discovery Education: Technology. Rosen Publishing Group, 2012.

[16] Ann M. Campbell, Michel Gendreau, and Barrett W. Thomas. The orienteering problem with stochastic travel and service times. Annals of Operations Research, 186(1):61–81, 6 2011.

[17] Carlos Carbone, Oscar Garibaldi, and Zohre Kurt. Swarm robotics as a solution to crops inspection for precision agriculture. KnE Engineering, 3(2):552–562, Feb. 2018.

[18] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. A fast and effective heuristic for the orienteering problem. European Journal of Operational Research, 88(3):475–489, 2 1996.

[19] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. The team orienteering problem. European Journal of Operational Research, 88(3):464–474, 2 1996.

[20] Roseneia Rodrigues Santos de Melo, Dayana Bastos Costa, Juliana Sampaio Álvares, and Javier Irizarry. Applicability of unmanned aerial system (uas) for safety inspection on construction sites. Safety Science, 98:174–185, 2017.

[21] Irina Dolinskaya, Zhenyu (Edwin) Shi, and Karen Smilowitz. Adaptive orienteering problem with stochastic travel times. Transportation Research Part E: Logistics and Transportation Review, 109:1–19, 1 2018.

[22] Lanah Evers, Kristiaan Glorie, Suzanne van der Ster, Ana Isabel Barros, and Herman Monsuur. A two-stage approach to the orienteering problem with stochastic weights. Computers and Operations Research, 43:248–260, 3 2014.

[23] Dominique Feillet, Pierre Dejax, and Michel Gendreau. Traveling salesman problems with profits. Transportation Science, 39(2):188–205, 2005.

[24] Avinash Gautam and Sudeept Mohan. A review of research in multi-robot systems. In 2012 IEEE 7th International Conference on Industrial and Information Systems (ICIIS), pages 1–5, 2012.

[25] Joshua Grass and Shlomo Zilberstein. Anytime Algorithm Development Tools. ACM SIGART Bulletin, 7(2):20–27, 1996.

[26] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. Orienteering Problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research, 255(2):315–332, 12 2016.

[27] Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R. Ravi. Approximation Algorithms for Stochastic Orienteering. In Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, page 1522–1538, 2012.

[28] Anupam Gupta, Ravishankar Krishnaswamy, Viswanath Nagarajan, and R Ravi. Running Errands in Time: Approximation Algorithms for Stochastic Orienteering. Mathematics of Operations Research, 40(1):56–79, 2015.

[29] Farouk Hammami, Monia Rekik, and Leandro C. Coelho. A hybrid adaptive large neighborhood search heuristic for the team orienteering problem. Computers & Operations Research, 123:105034, 11 2020.

[30] Pamela J. Hinds, Teresa L. Roberts, and Hank Jones. Whose job is it anyway? a study of human-robot interaction in a collaborative task. Human–Computer Interaction, 19(1-2):151–181, 2004.

[31] Taylan Ilhan, Seyed M R Iravani, and Mark S Daskin. The orienteering problem with stochastic profits. IIE Transactions, 40:406–421, 2008.

[32] Fabricio Rodrigues Inacio, Douglas G. Macharet, and Luiz Chaimowicz. Persistent monitoring of multiple areas of interest with robotic swarms. In 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), pages 82–87, 2018.

[33] Shafkat Islam and Abolfazl Razi. A path planning algorithm for collective monitoring using autonomous drones. 2019 53rd Annual Conference on Information Sciences and Systems (CISS), pages 1–6, 2019.

[34] Stefan Jorgensen, Robert H Chen, Mark B Milam, and Marco Pavone. The Team Surviving Orienteers problem: routing teams of robots in uncertain environments with survival constraints. In 2017 First IEEE International Conference on Robotic Computing (IRC), pages 227–234, 2017.

[35] Stefan Jorgensen, Robert H Chen, Mark B. Milam, and Marco Pavone. The Team Surviving Orienteers problem: routing teams of robots in uncertain environments with survival constraints. Autonomous Robots, 42(4):927–952, 2018.

[36] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime Motion Planning using the RRT *. In 2011 IEEE International Conference on Robotics and Automation, pages 1478–1483, 2011.

[37] Richard M. Karp. Reducibility among Combinatorial Problems. In Complexity of Computer Computations, pages 85–103. Springer US, Boston, MA, 1972.

[38] Ines Khoufi, Pascale Minet, and Nadjib Achir. Unmanned aerial vehicles path planning for area monitoring. In 2016 International Conference on Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN), pages 1–5, 2016.

[39] Gorka Kobeaga, María Merino, and Jose A. Lozano. An efficient evolutionary algorithm for the orienteering problem. Computers and Operations Research, 90:42–59, 2 2018.

[40] Gorka Kobeaga, María Merino, and Jose A Lozano. A revisited branch-and-cut algorithm for large-scale orienteering problems. ArXiv, abs/2011.0, 2020.

[41] Gilbert Laporte. The traveling salesman problem: An overview of exact and approximate algorithms. European Journal of Operational Research, 59(2):231–247, 6 1992.

[42] Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. Discrete Applied Mathematics, 26(2):193–207, 1990.

[43] Steven M LaValle. Planning Algorithms. Cambridge University Press, New York, NY, USA, 2006.

[44] Yang Li, Rongxin Cui, Weisheng Yan, and Demin Xu. Long-term adaptive informative path planning for scalar field monitoring using cross-entropy optimization. Science China Information Sciences, 62(5):1869–1919, 2019.

[45] Shih Wei Lin and Vincent F. Yu. Solving the team orienteering problem with time windows and mandatory visits by multi-start simulated annealing. Computers and Industrial Engineering, 114:195–205, 12 2017.

[46] Douglas G Macharet and Mario F M Campos. A survey on routing problems and robotic systems. Robotica, 36(12):1781–1803, 2018.

[47] Daniel B. Mesquita, Ronaldo F. dos Santos, Douglas G. Macharet, Mario F. M. Campos, and Erickson R. Nascimento. Fully convolutional siamese autoencoder for change detection in uav aerial images. IEEE Geoscience and Remote Sensing Letters, 17(8):1455–1459, 2020.

[48] V Papapanagiotou, R Montemanni, and L M Gambardella. Objective function evaluation methods for the orienteering problem with stochastic travel and service times. Journal of Applied Operational Research, 6(1):16–29, 2014.

[49] Lynne E Parker. Multiple Mobile Robot Systems, pages 921–941. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[50] Marija Popovic, Teresa Vidal-Calleja, Gregory Hitz, Inkyu Sa, Roland Siegwart, and Juan Nieto. Multiresolution Mapping and Informative Path Planning for UAV-based Terrain Monitoring. In IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1382–1388, 2017.

[51] Christopher Reardon and Jonathan Fink. Towards Joint Human-Robotic Solutions to Surveillance Problems. In International Florida Artificial Intelligence Research Society, 2017.

[52] Christopher Reardon, Fei Han, Hao Zhang, and Jonathan Fink. Optimizing autonomous surveillance route solutions from minimal human-robot interaction. In IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR), pages 208–215. IEEE, 10 2017.

[53] Lorena S. Reyes-Rubiano, Carlos F. Ospina-Trujillo, Javier Faulin, Jose M. Mozos, Javier Panadero, and Angel A. Juan. The Team Orienteering Problem With Stochastic Service Times and Driving-Range Limitations: A Simheuristic Approach. In 2018 Winter Simulation Conference (WSC), pages 3025–3035. IEEE, 12 2018.

[54] Javier Rodríguez-Robles, Álvaro Martin, Sergio Martin, José A. Ruipérez-Valiente, and Manuel Castro. Autonomous sensor network for rural agriculture environments, low cost, and energy self-charge. Sustainability, 12(15), 2020.

[55] Takuma Sakamoto, Stéphane Bonardi, and Takashi Kubota. A Routing Framework for Heterogeneous Multi-Robot Teams in Exploration Tasks. IEEE Robotics and Automation Letters, 5(4):6662–6669, 10 2020.

[56] Alberto Santini. An adaptive large neighbourhood search algorithm for the orienteering problem. Expert Systems with Applications, 123:154–167, 6 2019.

[57] Guangyao Shi, Lifeng Zhou, and Pratap Tokekar. Robust Multiple-Path Orienteering Problem: Securing Against Adversarial Attacks, 2020.

[58] Panagiotis Tampakis, Eva Chondrodima, Aggelos Pikrakis, Yannis Theodoridis, Kostis Pristouris, Harry Nakos, Eleni Petra, Theodore Dalamagas, Andreas Kandiros, Georgios Markakis, Irida Maina, and Stefanos Kavadas. Sea area monitoring and analysis of fishing vessels activity: The i4sea big data platform. In 2020 21st IEEE International Conference on Mobile Data Management (MDM), pages 275–280, 2020.

[59] Ervin Teng, Ceferino Gabriel Ramirez, and Bob Iannucci. Swatt: Synchronized wide-area sensing and autonomous target tracking. In 2018 IEEE International Symposium on Technologies for Homeland Security (HST), pages 1–7, 2018.

[60] Goran Tmušić, Salvatore Manfreda, Helge Aasen, Mike R. James, Gil Gonçalves, Eyal Ben-Dor, Anna Brook, Maria Polinova, Jose Juan Arranz, János Mészáros, Ruodan Zhuang, Kasper Johansen, Yoann Malbeteau, Isabel Pedroso de Lima, Corine Davids, Sorin Herban, and Matthew F. McCabe. Current practices in uas-based environmental monitoring. Remote Sensing, 12(6), 2020.

[61] Pratap Tokekar, Joshua Vander Hook, David Mulla, and Volkan Isler. Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture. IEEE Transactions on Robotics, 32(6):1498–1511, 12 2016.

[62] Pratap Tokekar, Joshua Vander Hook, David Mulla, Volkan Isler, Joshua Vander Hook, David Mulla, Volkan Isler, Joshua Vander Hook, David Mulla, and Volkan Isler. Sensor Planning for a Symbiotic UAV and UGV System for Precision Agriculture. IEEE Transactions on Robotics, 32(6):1498–1511, 11 2013.

[63] James P. Trevelyan, Sung-Chul Kang, and William R. Hamel. Robotics in Hazardous Applications, pages 1101–1126. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[64] Nikolaos Tsiogkas, Valerio De Carolis, and David M. Lane. Energy-constrained informative routing for AUVs. In OCEANS 2016 - Shanghai, pages 1–5. IEEE, 4 2016.

[65] Nikolaos Tsiogkas, Valerio De Carolis, and David M. Lane. Towards an online heuristic method for energy-constrained underwater sensing mission planning. In 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 6801–6808. IEEE, 9 2017.

[66] Nikolaos Tsiogkas and David M. Lane. An Evolutionary Algorithm for Online, Resource-Constrained, Multivehicle Sensing Mission Planning. IEEE Robotics and Automation Letters, 3(2):1199–1206, 4 2018.

[67] Nikolaos Tsiogkas and David M. Lane. DCOP: Dubins Correlated Orienteering Problem Optimizing Sensing Missions of a Nonholonomic Vehicle Under Budget Constraints. IEEE Robotics and Automation Letters, 3(4):2926–2933, 10 2018.

[68] Martijn Van Der Merwe, James P Minas, Melih Ozlen, and John W Hearne. The cooperative orienteering problem with time windows. Optimization Online, 7:11, 2014.

[69] Pieter Vansteenwegen and Aldy Gunawan. Orienteering Problems. EURO Advanced Tutorials on Operational Research. Springer International Publishing, Cham, 2019.

[70] Pieter Vansteenwegen, Wouter Souffriau, and Dirk Van Oudheusden. The orienteering problem: A survey. European Journal of Operational Research, 209(1):1–10, 2 2011.

[71] Jian Wang, Jiansheng Guo, Jicheng Chen, Shan Tian, and Taoyong Gu. Uncertain Team Orienteering Problem With Time Windows Based on Uncertainty Theory. IEEE Access, 7:63403–63414, 2019.

[72] Jian Wang, Jiansheng Guo, Mingfa Zheng, Zheng Murong, and Zhengxin Li. Research on a novel minimum-risk model for uncertain orienteering problem based on uncertainty theory. Soft Computing, 23:4573–4584, 2019.

[73] Jun Xu, Gürkan Solmaz, Rouhollah Rahmatizadeh, Damla Turgut, and Ladislau Bölöni. Animal monitoring with unmanned aerial vehicle-aided wireless sensor networks. In 2015 IEEE 40th Conference on Local Computer Networks (LCN), pages 125–132, 2015.

[74] Ala Eddine Yahiaoui, Aziz Moukrim, and M. Serairi. The clustered team orienteering problem. Computers and Operations Research, 111:386–399, 11 2019.

[75] Jingjin Yu, Mac Schwager, and Daniela Rus. Correlated Orienteering Problem and its application to informative path planning for persistent monitoring tasks. In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 342–349. IEEE, 9 2014.

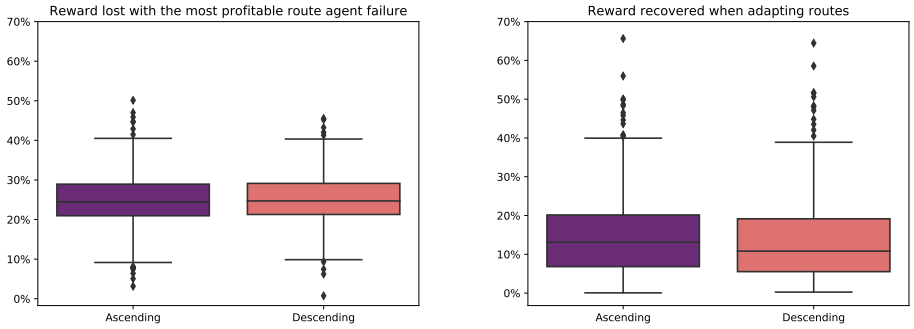
[76] Jingjin Yu, Mac Schwager, and Daniela Rus. Correlated Orienteering Problem and its Application to Persistent Monitoring Tasks. IEEE Transactions on Robotics, 32(5):1106–1118, 10 2016.

[77] Yue Zhang, Bin Song, Xiaojiang Du, and Mohsen Guizani. Vehicle tracking using surveillance with multimodal data fusion. IEEE Transactions on Intelligent Transportation Systems, 19(7):2353–2361, 2018.

Appendix A

Heterogeneous perception

This appendix shows charts and analyses that contributed to the conclusions outlined in Section 5.2.2. Regarding the order of choice of agents to set an initial route at a level in the region-level planning stage, we considered two strategies: ordering the agents by sensor information quality in ascending or descending order. When placing an initial route, the ascending strategy first chooses the agents with the sensors that provide the lowest information quality. In contrast, the descending approach starts this process with the agents that capture information with more quality.



(a) (b)

Figure A.1: Percentages of rewards lost with the failure and recovered with the adapting routine. (a) Percentage lost on the robot's failure responsible for the most profitable route; (b) Percentage recovered after failure using the proposed adaptation method.

From Figure A.1a, we can see that in the general case, the order of choice of agents does not make a difference in relation to the percentage of the planned reward lost due to the failure. The results are also very similar in Figure A.1b. Still, looking closer, we see that the median of the ascending strategy results is slightly higher than that of the descending strategy, and the box limits also follow this slight increase in recovered reward. Due to this observation, we considered the ascending strategy for ordering agents by the quality of information collected by their sensor in Section 5.2.2 discussion. In Figure A.1b, some outliers indicate a reward recovery greater than what was lost according to Figure A.1a. It may seem strange, but this really can happen. Just consider that a robot with a better-quality sensor will adapt its route to visit some POIs discovered by a teammate with a worse sensor that failed.

To improve the reading, we replicated Table 5.7, which shows the robots considered in the heterogeneous perception experiments, in Table A.1.

Table A.1: Considered robots in the experiments with heterogeneous perception. The classification was given according to the robot that can carry the sensor in the example.

Robot ID	Robot class	Information quality	Correlation range	Sensor example
HWINGS	A-FW	◐○○○	●●●●	RGB camera (at 100m)
LWINGS	A-FW	●○○○	●●●○	Thermal camera (at 60m)
QUAD*	A-MR	●●○○	●●○○	Multispectral camera (at 40m)
TRACKS	G-RT	●●●○	●◐○○	RGB-D camera
LEGS	G-RL	●●●●	●○○○	Soil moisture and temperature

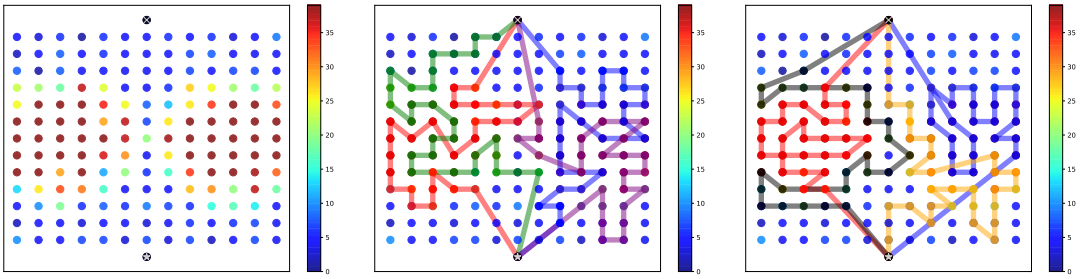
* Standard Robot configuration.

To deepen the results study, we list some specific teams to highlight some cases:

- TEAM-1: team with homogeneous robots. This team allows us to have a comparison parameter against the homogeneous version of the problem discussed in Chapter 4.
  Team composition: QUAD-1 QUAD-2 QUAD-3 QUAD-4
- TEAM-2: team with three equal robots and one robot with a sensor that captures information with 50% more quality than the others. The correlation range of this sensor is half of the others.
  Team composition: QUAD-1 QUAD-2 QUAD-3 LEGS-1
- TEAM-3: team with two pairs of equal robots. One of the pairs of robots has sensors that capture information with 25% more quality and has a correlation range of 25% less than the robots in the other pair.
  Team composition: QUAD-1 QUAD-2 TRACKS-1 TRACKS-2
- TEAM-4: team where all robots have different perception capabilities. Compared with the SR robot in the team, one of the robots has a sensor 50% worst in quality with a correlation range 50% higher. The other two robots have higher-quality sensors but with smaller correlation ranges. One is 25% more quality and 25% less range than SR; the other is 50% more quality and 50% less correlation range.
  Team composition: QUAD-1 LWINGS-1 TRACKS-1 LEGS-1

Figure A.2a shows an environment composed of 169 POIs arranged in a grid and with two regions more important to be inspected in the environment central region. The routes plotted in Figure A.2b were generated for a team where all the robots have the same perception capabilities. As discussed in Chapter 4, the routes intertwine in the most

important regions of the environment to benefit the route adaptation routine in case of any agent failure. In Figure A.2c, we have the routes generated for the TEAM-3. The red and blue paths are for robots QUAD-1 and QUAD-2, respectively, while the orange and black routes are for robots TRACKS-1 and TRACKS-2. We see that these routes are slightly less intertwined than the routes of the homogeneous team and that the robots with sensors that collect information with lower quality have more responsibility to sense the regions more important to inspect. Robots with better sensors but with a smaller correlation range have routes bordering critical areas and can cover the region in case of a partner's failure. At first, this strategy may seem counterintuitive, as if robots with higher quality sensors were responsible for the most important regions of the environment, the reward collected at the end would be much greater. However, assuming that a failure will occur, the planned reward loss for the team could be very drastic in these cases. As we aim to reduce the impact of failures and generate routes that benefit the real-time adaptation routine, sending agents with lower-quality sensors to these regions makes more sense. If these agents fail, the teammates will adapt the routes and collect the information that would be lost. If they do not fail, coverage of these regions will be guaranteed, even with a lower quality.



(a) Environment (b) TEAM-1 (c) TEAM-3

Figure A.2: Comparison of routes generated for a homogeneous team and for a heterogeneous team. (a) Environment representation; (b) routes generated for a homogeneous team; (c) routes generated for a heterogeneous team.

To further the understanding of generated routes, we look at the relation between the number of points on the routes versus the reward collected. Figure A.3 compares the three picked heterogeneous teams. In this case, there is no direct relationship between the number of POIs and the planned reward. Since all robots have the same budget, the routes have, on average, the same amount of points to be inspected. However, the plots in the figure clarify the priority given to robots with poorer-quality sensors. On average, the routes of those with sensors that capture better quality information yield a lower reward, indicating that those robots visit more POIs outside critical regions. This strategy tries to balance out the collected rewards and mitigate the impact of possible agent failures.

We also compared the performance of these four spotlighted teams when the robot responsible for the most profitable route in the plan failed right at the beginning of the

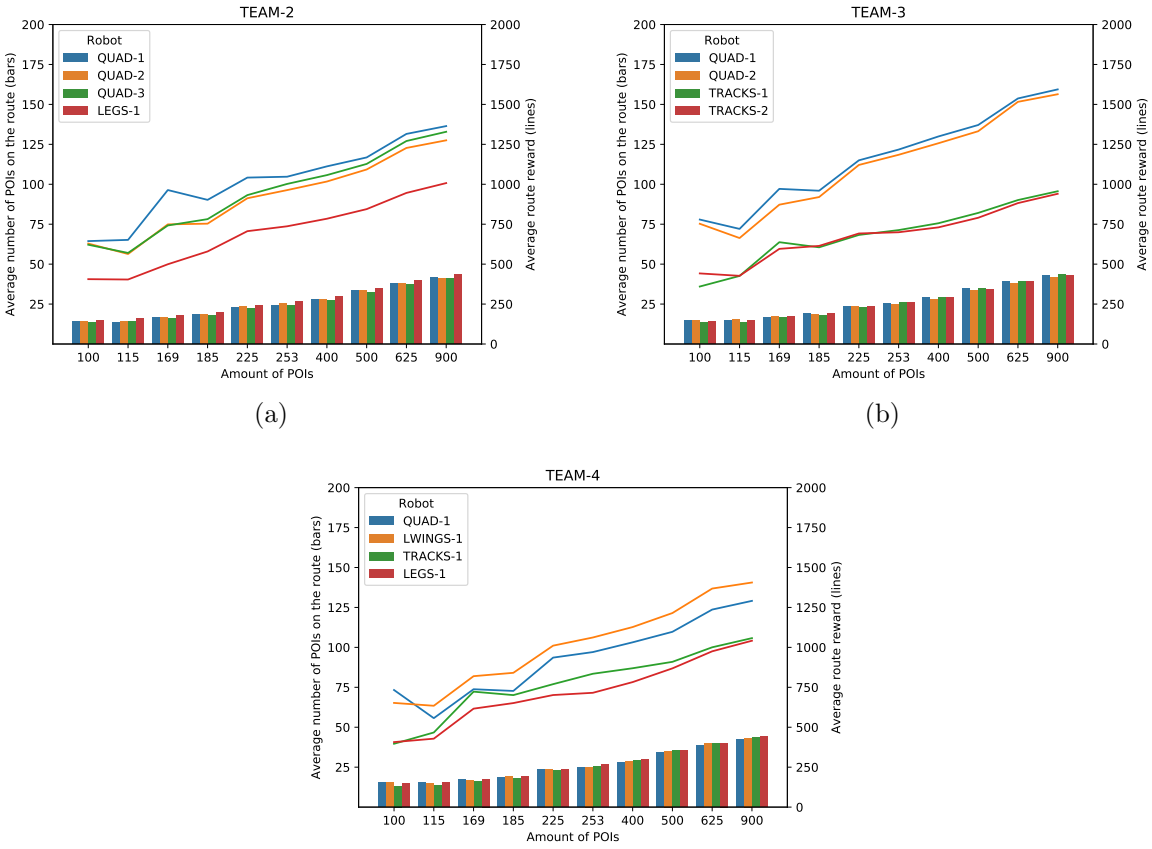


Figure A.3: Analysis of the routes generated for the picked heterogeneous teams. The average amount of POIs on each route is indicated with the bars (left axis), and the average reward of the routes is represented by the lines (right axis).

mission. In the first plot of Figure A.4, we see the average reward planned according to the instances' number of POIs. We highlight TEAM-4, which is the most heterogeneous team in the group. In this team, in addition to SR, we have LWINGS-1 with a sensor with a quality 25% lower than SR, and robots TRACKS-1 and LEGS-1 with 25% and 50% more quality in your sensors compared to SR, respectively. The presence of LWINGS-1 in the team greatly affects the reward planned since it will be responsible for the most important POIs to be inspected. For example, the performance of TEAM-2, which has only one robot LEGS-1 and three SR, is considerably better because three agents are responsible for the important POIs, and they carries better quality sensors than LWINGS-1. In the second plot, we have the result when teammates ignore their partner's failure. The TEAM-4 has the worst performance among the highlighted teams just because it had only one agent responsible for the important POIs. This no happens with TEAM-2, where the most important points to be inspected were divided between the three agents of type QUAD. The last plot shows the result of the route adaptation routine after the partner's failure. The TEAM-4 has the highest reward recovery because it has agents with better-quality sensors that will adapt their routes. But we see that the final results of all teams are not

much different. In general, a team like TEAM-3 would be the best option when dealing with different perceptions. Due to the team being balanced with two agents of each type, the initial plan does not overload a single agent, and when there is a failure, the loss is not so drastic. In addition, the result of adapting routes also remains high.

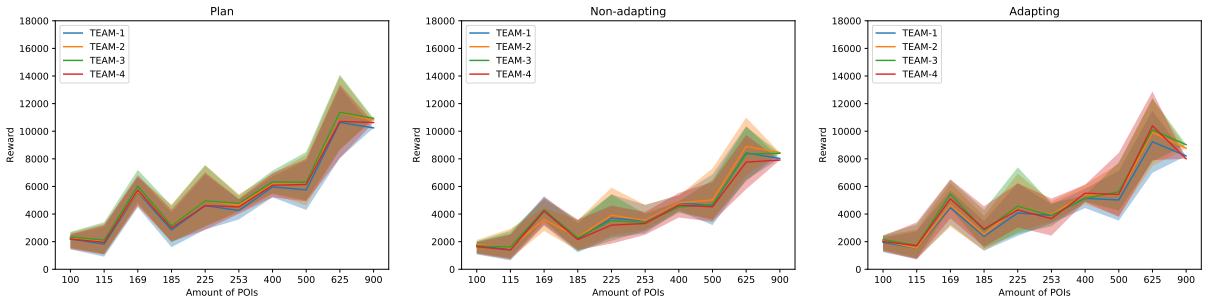
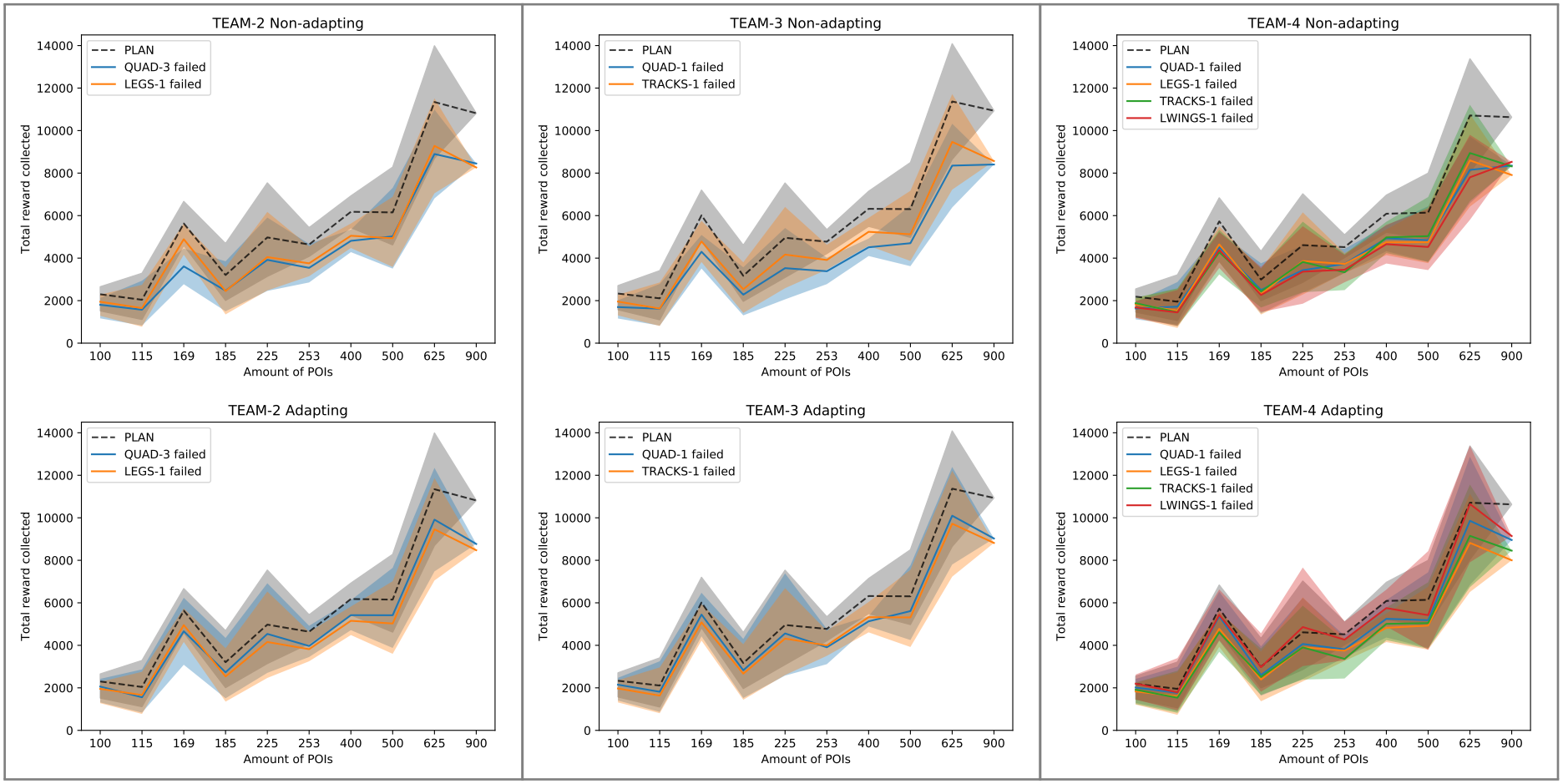


Figure A.4: Comparison of the computed average rewards considering the four spotlighted teams, according to the number of POIs in the instance.

The results in Figure A.4 show the failure of the robot that would follow the plan's most profitable route, regardless of the type of robot. In Figure A.5, we compare when the failure occurs according to the type of robot better to understand the impact of each one on the mission. In these plots, for each team, we selected among the robots of the same type the one responsible for the most profitable route and simulated its failure at the beginning of the mission. The black dashed lines represent the average planned reward for the team. At the top of Figure A.5a, regardless of the failed agent, the impact on the reward collected was not that different when teammates ignored the failure. Adapting the agents' routes, in the bottom plot, we see that when QUAD-3 fails, the LEGS-1 adapted its route to cover part of the POIs not visited by QUAD-3 and achieved an average reward collected higher than when LEGS-1 fails. This considerable reward recovery is also due to the LEGS-1 sensor that collects information with a 50% higher quality than the sensor of QUAD-3.

Figure A.5b shows the results of a team composed of two agents with sensors that capture information with 25% more quality and a correlation range 25% smaller than the other two. When TRACKS-1 fails, and the other robots keep their paths, the reward collected is greater than when QUAD-2 fails. This is expected since we prioritize agents with lower quality sensors, so QUAD type robots are responsible for inspecting the most valuable POIs, and the failure of TRACKS-1 does not impact the result so much. However, adapting the agents' routes after the failure shows similar results regardless of the failed robot. That's because when QUAD-2 fails, the two TRACKS-type robots on the team collect a reward 25% higher than what would be collected by QUAD-2. The scenario where all robots have different sensors is plotted in Figure A.5c. When agents ignore the partner failure, the greatest reward loss is when agents with the lowest quality sensors and the highest correlation range fail. However, this scenario is reversed when



(a) (b) (c)

Figure A.5: Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance. For each type of agent on the team, we force the failure of the one responsible for the most profitable route. The upper plots show when the team members ignore the partner's failure, and the lower when the agents adapt their routes. The black dashed line corresponds to the planned reward for the team.

robots adapt their routes after failure. When a robot with an inferior sensor fails, the rest of the team will collect the reward from the POIs with an extra percentage because of the sensor quality. In some cases, the average reward collected at the end of the mission is even higher than planned, as we see in instances with 225 POIs when LWINGS-1 fails. This can happen because the agents still active on the mission have sensors 25%, 50%, and 100% better than the sensor carried by LWINGS-1. These better sensors greatly increase the reward collected in regions with more important POIs to inspect uncovered by the LWINGS-1 failure. But why not send these agents to the critical areas already in the planning phase? Precisely because we don't know which agent will fail and when this will happen. Our method tries to reduce the impacts of failures that can occur at any time, and for that, we have seen that it is best to leave the agents with the worst sensors responsible for the most critical regions.

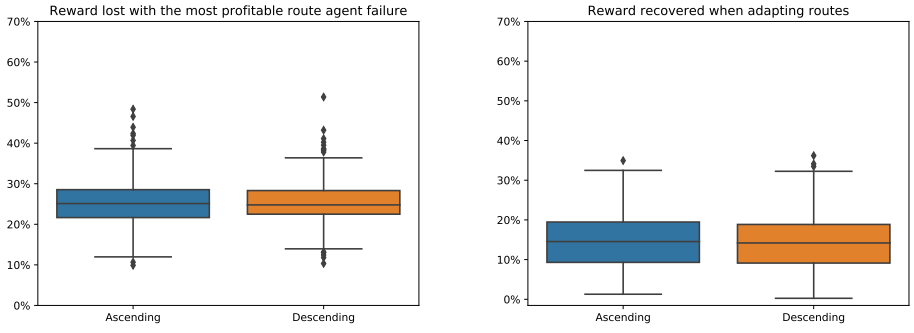
Appendix B

Heterogeneous costs

This Appendix shows charts and analyses that contributed to the conclusions outlined in Section 5.2.3. As in the mentioned Section, we divided this Appendix into three subsections: heterogeneous speeds, heterogeneous sensing times, and finally, we considered both heterogeneous costs at the team.

B.1 Different speeds

We consider two strategies in setting the initial routes at a level in the region-level planning phase: ordering the agents by the maximum speed in ascending or descending order. The ascending strategy first chooses the slowest agents, while the descending approach starts this process with the fastest agents. Figure B.1 compares the collected reward percentage in failure and route adaptation cases.



(a) (b)

Figure B.1: Percentages of rewards lost with the failure and recovered with the adapting routine. (a) Percentage lost on the robot's failure responsible for the most profitable route; (b) Percentage recovered after failure using the proposed adaptation method.

Figure B.1a shows the percentage of the planned reward lost when the agent responsible for the most profitable route fails at the beginning of the mission. We noticed that when considering the descending strategy, the percentage of reward lost with fail-

ures varies slightly less than when considering the ascending order. Figure B.1b is more difficult to conclude, but considering the ascending order has a slight advantage, that is, prioritizing the slowest agents in the planning phase has a slight advantage when it comes to recovering part of the reward that would be lost with the failure of an agent in the mission execution. Since our proposal is to use our planning method in conjunction with our route adaptation routine and considering what was analyzed in the previous Sections, we chose the ascending order as the best option for ordering the agents in the region-level planning. In this way, we continue the analysis prioritizing the slowest agents of the team in the two analyzed planning stages.

To improve the reading, we replicated Table 5.8, which shows the robots considered in the heterogeneous speed experiments, in Table B.1.

Table B.1: Considered robots in the experiments with heterogeneous speeds. The classification was given exclusively about maximum speed.

Robot ID	**Robot class**	**Maximum speed**	**Sensing time**
TRACKS	Ground Robot Tracked (G-RT)	●○○○	●●○○
WHEELS*	Ground Robot Wheeled (G-RW)	●●○○	●●○○
QUAD	Aerial Multi-Rotor (A-MR)	●●●○	●●○○
WINGS	Aerial Fixed-Wing (A-FW)	●●●●	●●○○

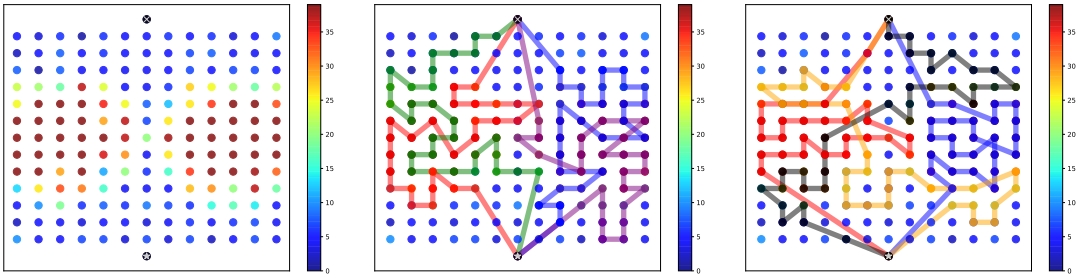
* Standard Robot configuration.

To deepen the analysis, we list some specific teams to highlight some cases:

- TEAM-1: team with homogeneous robots. This team allows us to have a comparison parameter against the homogeneous version of the problem discussed in Chapter 4.
  Team composition: WHEELS-1 WHEELS-2 WHEELS-3 WHEELS-4
- TEAM-2: team with three equal-speed robots, while one was twice as fast.
  Team composition: WHEELS-1 WHEELS-2 WHEELS-3 WINGS-1
- TEAM-3: team with two pairs of equal robots. One of the pairs of robots is 50% faster than the robots in the other pair.
  Team composition: WHEELS-1 WHEELS-2 QUAD-1 QUAD-2
- TEAM-4: team where all robots have different speeds. One robot has half the speed of SR while the other two are 50% and 100% faster than SR.
  Team composition: WHEELS-1 QUAD-1 WINGS-1 TRACKS-1

Figure B.2a shows an instance with two regions more important to be inspected in the central region. Figure B.2b is the plan generated for the team where all the robots

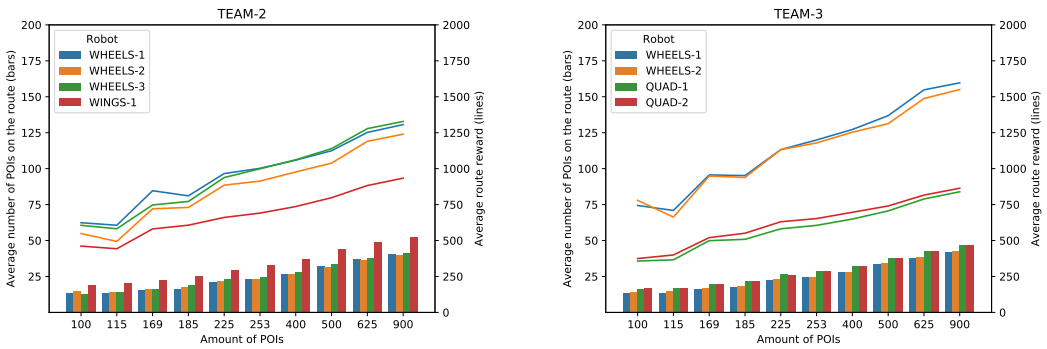
have the same speed. The routes are more intertwined in the central region to benefit the route adaptation routine in case of any agent failure. In Figure B.2c, we have the routes generated for the TEAM-3. The red and blue paths are for robots WHEELS-1 and WHEELS-2, respectively, while the orange and black routes are for robots QUAD-1 and QUAD-2. In these routes, the slower robots have more responsibility to sense the regions more important to inspect. The fastest robots have routes bordering critical areas and can cover the region in case of a partner's failure. These characteristics were also observed in the routes of the other heterogeneous teams.



(a) Environment (b) TEAM-1 (c) TEAM-3

Figure B.2: Comparison of routes generated for a homogeneous team and for a heterogeneous team. (a) Environment representation; (b) routes generated for a homogeneous team; (c) routes generated for a heterogeneous team.

We also examine the relation between the number of points on the routes versus the reward collected. Figure B.3 compares the three picked heterogeneous teams. On average, the fastest robots visited more points but collected less reward. This was expected because if we have a budget measured in time, a faster robot with the same budget can visit more points than a slower robot. However, the difference in the number of points visited by fast and slow robots was not as significant as when we compared teams with different budgets in Section 5.2.1.



(a) (b)

Figure B.3: Analysis of the routes generated for the picked heterogeneous teams. The average amount of POIs on each route is indicated with the bars (left axis), and the average reward of the routes is represented by the lines (right axis).

Figure B.4 compares the four spotlighted teams when the robot responsible for the most profitable route in the plan failed right at the beginning of the mission. The team with homogeneous agents in the first plot has a slightly lower result than the others in the initial plan. But there is not a big difference between planned rewards. In the second plot, we see that teams composed of agents with different speeds have a greater loss of reward when teammates ignore their partner's failure. This reflects our planning method, which generates routes prioritizing slower agents. However, when using our route adaptation routine in the third plot, heterogeneous teams benefit from the initial plan, making it possible to retrieve more information than if all agents had the same speed.

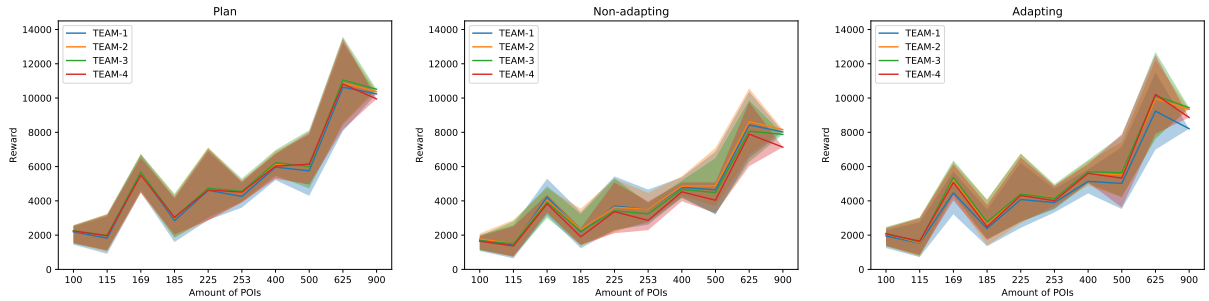
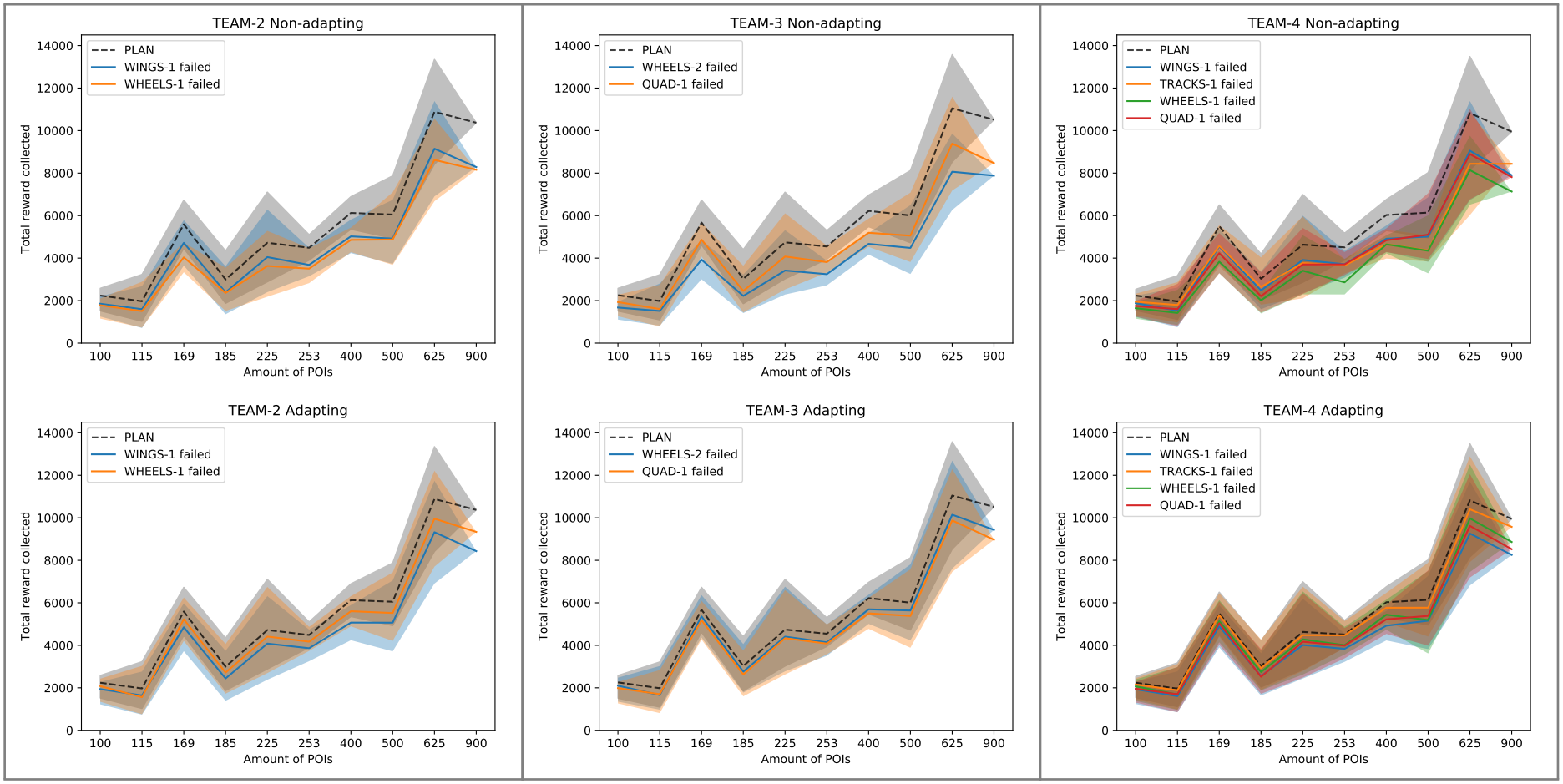


Figure B.4: Comparison of the computed average rewards considering the four spotlighted teams, according to the number of POIs in the instance.

The results in Figure B.4 show the failure of the robot that would follow the plan's most profitable route, regardless of the type of robot. In Figure B.5, we compare when the failure occurs according to the type of robot better to understand the impact of each agent on the mission. In these plots, for each team, we selected among the robots of the same type the one responsible for the most profitable route and simulated its failure at the beginning of the mission. The black dashed lines represent the average planned reward for the team. The TEAM-2 team has three WHEELS-type robots and one WINGS-type robot, which is twice as fast as the others. We see in Figure B.5a that when WINGS-1 fails and the other agents do nothing, the reward collected by the team is greater than when WHEELS-1 fails. However, this is reversed when we use our adaptation routine in the bottom plot of Figure B.5a. When WHEELS-1 fails, we have that WINGS-1 will have its route adapted and, because it is faster, it can cover a greater number of points and recoup much of the reward that would have been lost by failure.

When the team has two pairs of agents with the same speed, the difference in reward lost depending on the failed agent is greater. In Figure B.5b, when the other robots ignore the partner's failure, the reward collected at the end is greater when the fastest robot fails. Since we prioritize the slower agents when planning, WHEELS type robots are responsible for visiting the most valuable POIs, and when one of these robots fails, the loss is greater. However, adapting the agents' routes after the failure shows similar results, regardless of the failed robot. This is because when the WHEELS-2 fails,



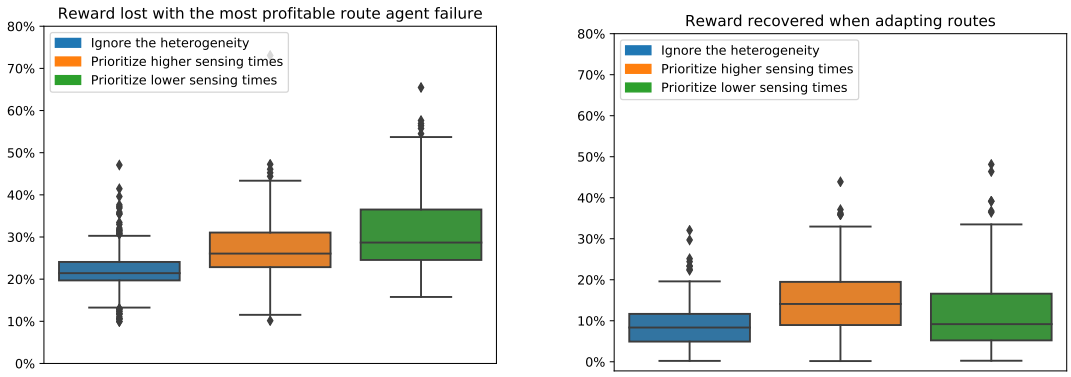
(a) (b) (c)

Figure B.5: Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance. For each type of agent on the team, we force the failure of the one responsible for the most profitable route. The upper plots show when the team members ignore the partner's failure, and the lower when the agents adapt their routes. The black dashed line corresponds to the planned reward for the team.

the two QUAD-type robots on the team, which are 50% fastest, adapt their routes and cover the points not visited by WHEELS-2. In Figure B.5c, we have a team where all robots have different speeds. When agents ignore the teammate's failure, the smallest reward loss occurs when WINGS-1 or QUAD-1 fail. However, we see that the loss with the WHEELS-1 failure is greater than the TRACKS-1 failure, which is a robot 50% slower. As mentioned, the TRACKS-1 is so slow that, even having priority to inspect the most valuable POIs, its budget does not allow it to inspect many points. When the robots adapt their routes after the teammate's failure, the faster the active agents, the greater the reward collected. We emphasize the case when TRACKS-1 fails, and the active agents in the mission recover a large portion of the reward that the team would lose.

B.2 Different sensing times

To better understand the results and be able to indicate the best strategy used as criteria by the SELECTAGENT function, Figure B.6 shows the results obtained for the planning and simulation considering all the teams in a different chart.



(a) (b)

Figure B.6: Percentages of rewards lost with the failure and recovered with the adapting routine. (a) Percentage lost on the robot's failure responsible for the most profitable route; (b) Percentage recovered after failure using the proposed adaptation method.

Figure B.6a shows that ignoring the heterogeneity is still a good idea when the other agents ignore the partner failure. But we also see that prioritizing agents taking less time to sense a point is the worst alternative. In Figure B.6b, we see that prioritizing agents with higher sensing time is the strategy that most retrieves information using our adaptation routine. Therefore, considering different sensing times, we prioritized those agents that spent the most time collecting information from the environment.

Regarding the order of agents to set an initial route at a level in the region-level planning, Figure B.7 compares the collected reward percentage in failure and route adaptation cases. Analyzing the plots does not make the best strategy so clear. Looking closely, we can see that the median result of the descending approach is slightly bigger and that the boxes are the same size. Considering this slight superiority and following the pattern of other heterogeneities where the ordering of agents in this process followed the priority defined in the SELECTAGENT, we adopted the descending ordering of agents in the region-level planning as the best strategy for general cases considering agents with different sensing times.



(a) (b)

Figure B.7: Percentages of rewards lost with the failure and recovered with the adapting routine. (a) Percentage lost on the robot's failure responsible for the most profitable route; (b) Percentage recovered after failure using the proposed adaptation method.

To improve the reading, we replicated Table 5.9, which shows the robots considered in the heterogeneous sensing times experiments, in Table B.2.

Table B.2: Considered robots in the experiments with heterogeneous sensing times. The classification was given exclusively about the type of agent that could take that time to sense a point.

Robot ID	**Robot class**	**Maximum speed**	**Sensing time**
WINGS	Aerial Fixed-Wing (A-FW)	●●○○	●○○○
QUAD*	Aerial Multi-Rotor (A-MR)	●●○○	●●○○
WHEELS	Ground Robot Wheeled (G-RW)	●●○○	●●●○
LEGS	Ground Robot Legged (G-RL)	●●○○	●●●●

* Standard Robot configuration.

Once we define the strategy that best fits this heterogeneity in the general case, we will highlight some teams for a deeper analysis of the results:

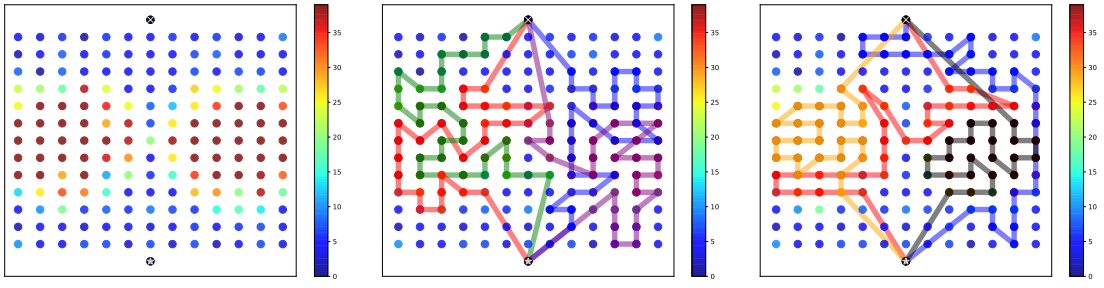
- TEAM-1: team with homogeneous robots. This team allows us to have a comparison parameter against the homogeneous version of the problem discussed in Chapter 4.
  Team composition: QUAD-1 QUAD-2 QUAD-3 QUAD-4

- TEAM-2: team with three equal robots and one robot that spends twice more time sensing the environment as the others.
  Team composition: QUAD-1 QUAD-2 QUAD-3 LEGS-1

- TEAM-3: team with two pairs of equal robots. One of the pairs of robots spends 50% more time inspecting a point than the robots in the other pair.
  Team composition: QUAD-1 QUAD-2 WHEELS-1 WHEELS-2

- TEAM-4: team where all robots have different sensing times. One robot spends half the time than the SR to collect data. The other two agents are 50% and 100% faster than SR to sense the environment.
  Team composition: QUAD-1 WINGS-1 WHEELS-1 LEGS-1

Figure B.8a shows an instance with two regions more important to be inspected in the central region. Figure B.8b shows the routes planned for the team when all the robots have the same sensing time. In Figure B.8c, we have the routes generated for the TEAM-3. The red and blue paths are for robots QUAD-1 and QUAD-2, respectively, while the orange and black routes are for robots WHEELS-1 and WHEELS-2. In these routes, the robots with higher sensing costs are the most responsible for sensing the regions more important to inspect. The robots that take less time to capture information have routes bordering critical areas and can cover the region in case of a partner's failure. These characteristics were also observed in the routes of the other heterogeneous teams.



(a) Environment (b) TEAM-1 (c) TEAM-3

Figure B.8: Comparison of routes generated for a homogeneous team and for a heterogeneous team. (a) Environment representation; (b) routes generated for a homogeneous team; (c) routes generated for a heterogeneous team.

We also examine the relation between the number of points on the routes versus the reward collected. Figure B.9 compares the three picked heterogeneous teams. On average, the robots that spend more time sensing a POI have routes with fewer points,

but they are the most profitable routes in the plan. Indeed, spending more time inspecting a point will result in less time to move between POIs. But we noticed that the difference between the number of POIs on the routes is less significant than when we consider agents with different speeds in the previous section, which indicates that the sensing time has less impact on planning than the speed of the robots.

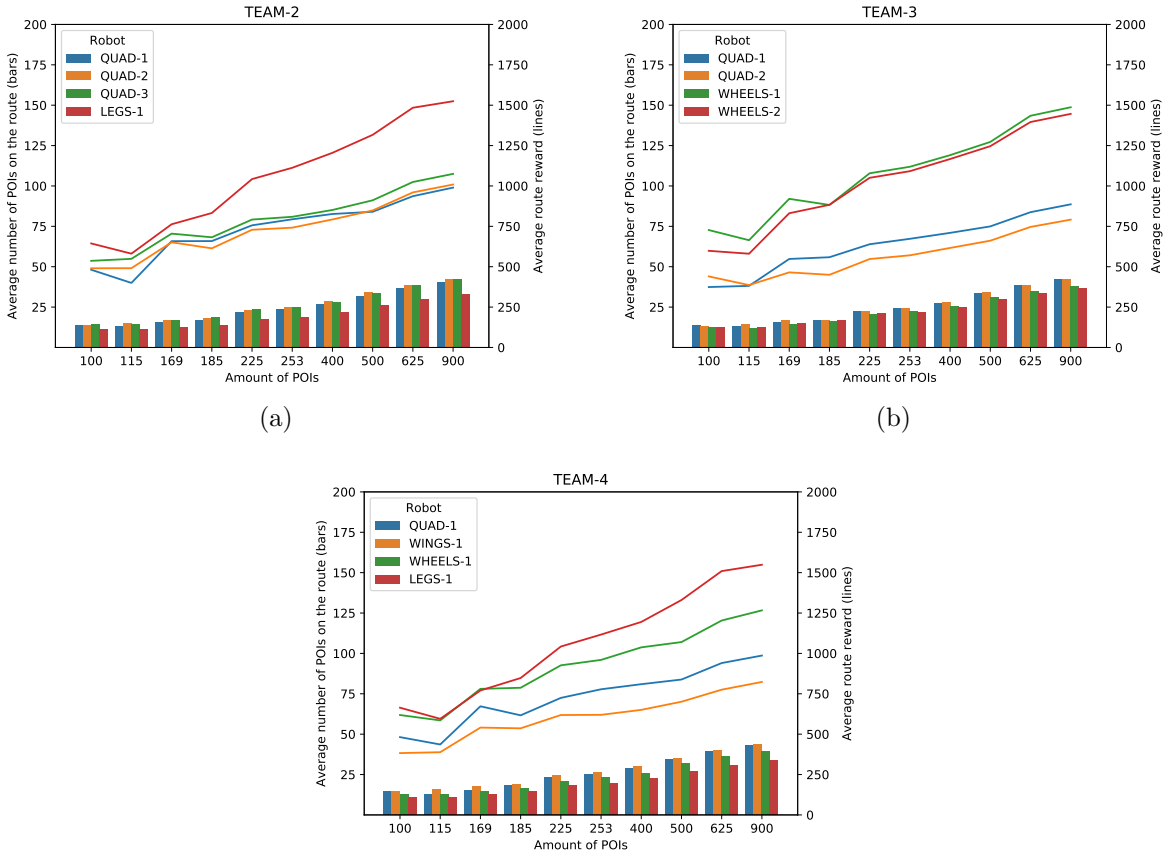


Figure B.9: Analysis of the routes generated for the picked heterogeneous teams. The average amount of POIs on each route is indicated with the bars (left axis), and the average reward of the routes is represented by the lines (right axis).

Figure B.10 compares these four spotlighted teams in simulations where the robot responsible for the most profitable route in the plan failed right at the beginning of the mission. There is almost no difference between the average rewards planned in the first plot, regardless of the team for which the plan was made. In the second plot, we see that teams composed of agents with different sensing times have a greater loss of reward when teammates ignore their partner's failure. This reflects our planning method, which generates routes prioritizing slower agents to sense. However, when using our route adaptation routine in the third plot, heterogeneous teams benefit from the initial plan, making it possible to retrieve more information than if all agents had the same sensing time. We can see that when robots adapt their routes, the average reward collected is

similar regardless of the team, as observed in the case of planned rewards in the first plot.

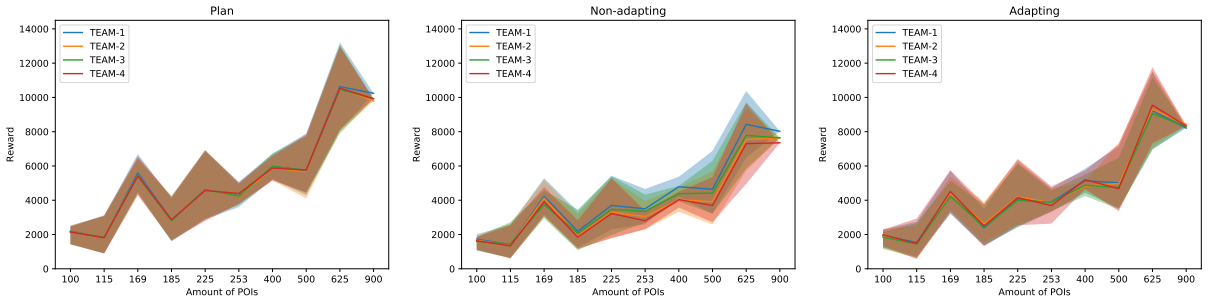
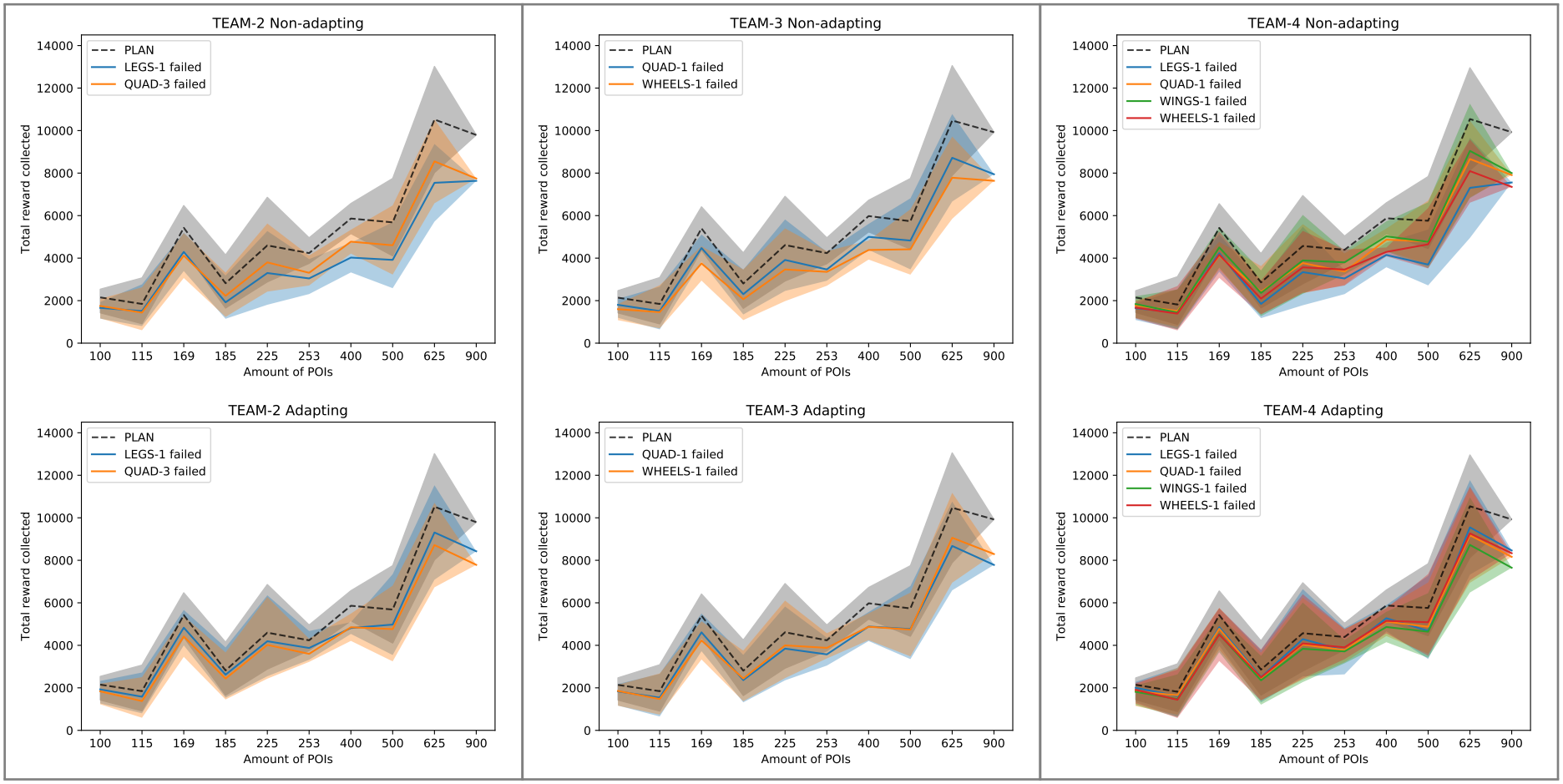


Figure B.10: Comparison of the computed average rewards considering the four spotlighted teams, according to the number of POIs in the instance.

The results in Figure B.10 show the failure of the robot that would follow the plan's most profitable route, regardless of the type of robot. In Figure B.11, we compare when the failure occurs according to the type of robot better to understand the impact of each agent on the mission. In these plots, for each team, we selected among the robots of the same type the one responsible for the most profitable route and simulated its failure at the beginning of the mission. The black dashed lines represent the average planned reward for the team.

The TEAM-2 team has three QUAD-type robots and one LEGS-type robot, which spends twice as much time to sense a point. Figure B.11a shows that when LEGS-1 fails, and the other agents do nothing, the reward lost is greater than when QUAD-3 fails. That's because LEGS-1 was prioritized in the planning phase and is responsible for the team's most profitable route. However, when we use our adaptation routine in the bottom plot of Figure B.11a, we see that the reward collected when LEGS-1 fails is greater, as the three QUAD-type agents spend less time to sense a POI and can cover the discovered ones. The scenario is similar when the team has two pairs of agents with the same sensing times. In Figure B.11b, when the other robots ignore the partner's failure, we have a greater reward loss when a robot with a longer sensing time fails because we prioritize them in the planning phase. However, adapting the agents' routes after the failure makes the reward collected more equal in both cases. In Figure B.11c, we have a team where all robots have different sensing times. When agents ignore the teammate's failure, the reward lost grows according to the sensing time of failed robot. That is, the greater the sensing time of the agent that fails, the greater the team's loss of reward. This scenario is reversed when the robots adapt their routes after the teammate's failure. The more active agents with shorter sensing time adjusting their routes, the greater the reward the team recovers.

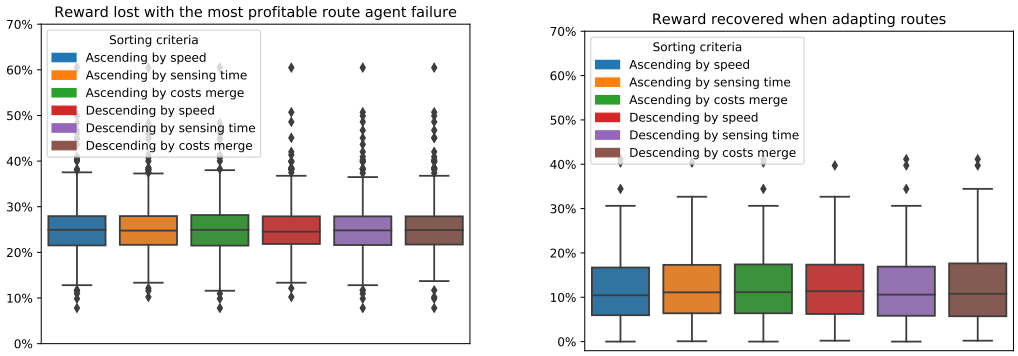


(a) (b) (c)

Figure B.11: Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance. For each type of agent on the team, we force the failure of the one responsible for the most profitable route. The upper plots show when the team members ignore the partner's failure, and the lower when the agents adapt their routes. The black dashed line corresponds to the planned reward for the team.

B.3 Different speeds and sensing times

Figure B.12 compares the collected reward percentage in failure and route adaptation cases considering all sorting criteria used to set an initial route at a level in the region-level planning stage. Analyzing the averages of loss and recovery, we concluded that the best ordering criterion is ascending by sensing time. With this criterion, the team loses an average of 25.6% of the reward with the agent's failure and manages to recover an average of 11.1% of the planned reward that would be lost.



(a) (b)

Figure B.12: Percentages of rewards lost with the failure and recovered with the adapting routine. (a) Percentage lost on the robot's failure responsible for the most profitable route; (b) Percentage recovered after failure using the proposed adaptation method.

To recap, the best general strategy when considering various speeds and sensing times is to prioritize slower agents when defining which agent has priority in visiting a POI. Furthermore, when setting an initial route at a level in the region-level planning stage, it is better to order agents by sensing time and choosing the fastest ones first. To improve the reading, Table B.3 shows again the robots considered in this experiment.

Table B.3: Considered robots in the experiments with heterogeneous costs. The classification was given exclusively about maximum speed.

Robot ID	**Robot class**	**Maximum speed**	**Sensing time**
LEGS	Ground Robot Legged (G-RL)	●○○○	●○○○
TRACKS	Ground Robot Tracked (G-RT)	●○○○	●●●●
WHEELS*	Ground Robot Wheeled (G-RW)	●●○○	●●○○
QUAD	Aerial Multi-Rotor (A-MR)	●●●○	●○○○
HEXA	Aerial Multi-Rotor (A-MR)	●●●○	●●●○

* Standard Robot configuration.

For a deeper analysis, we pick some teams composed of robots from Table B.3. We changed the pattern of chosen teams in this analysis to be able to analyze a larger group of combinations between different speeds and sensing times. We picked the following teams:

- TEAM-1: team with homogeneous robots. This team allows us to have a comparison parameter against the homogeneous version of the problem discussed in Chapter 4. Team composition: WHEELS-1 WHEELS-2 WHEELS-3 WHEELS-4
- TEAM-2: team with two pairs of equal robots. One of the pairs of robots is 50% faster and takes 50% less time to sense a POI than the robots in the other pair. Team composition: WHEELS-1 WHEELS-2 QUAD-1 QUAD-2
- TEAM-3: team with two pairs of equal robots. One of the pairs of robots is 50% faster and takes 50% more time to sense a POI than the robots in the other pair. Team composition: WHEELS-1 WHEELS-2 HEXA-1 HEXA-2
- TEAM-4: team with two pairs of equal robots. One of the pairs of robots is 50% slower and takes twice more time to sense a POI than the robots in the other pair. Team composition: WHEELS-1 WHEELS-2 TRACKS-1 TRACKS-2
- TEAM-5: team with two pairs of equal robots. One of the pairs of robots is 50% slower and takes 50% less time to sense a POI than the robots in the other pair. Team composition: WHEELS-1 WHEELS-2 LEGS-1 LEGS-2

With these teams, we examined the relationship between the number of points on the routes versus the collected reward. Figure B.13 compares the four chosen heterogeneous teams. As we prioritize the slowest agents, the routes generated are, on average, more profitable for these robots, even with fewer points. This had already been observed when we analyzed the speed separately. But here, we also have the impact of different sensing times. This impact is clear in Figures B.13a and B.13b where the difference in the number of points on the routes is less significant than when we analyze only the speed. We can also see the impact of sensing times in Figures B.13c and B.13d. In these cases, the team has two robots with half the speed of the SR, which have priority to inspect the POIs. In this way, the robots of type TRACKS and LEGS should have the most valuable routes. We observed that this does not happen because they are very slow, which ends up interfering with the number of points they manage to inspect within the given budget. However, we see that TRACKS-type robots in Figure B.13c have a different pattern than LEGS-type robots in Figure B.13d. This is due to the sensing time of each robot, which further limits the number of points that TRACKS-type robots can visit. TRACKS-type robots spend four times more time to sense a point than LEGS-type ones.

Figure B.14 compares the five picked teams when the robot responsible for the most profitable route failed right at the beginning of the mission. The teams with two SR and two faster agents obtained the highest planned reward, while the teams with robots slower

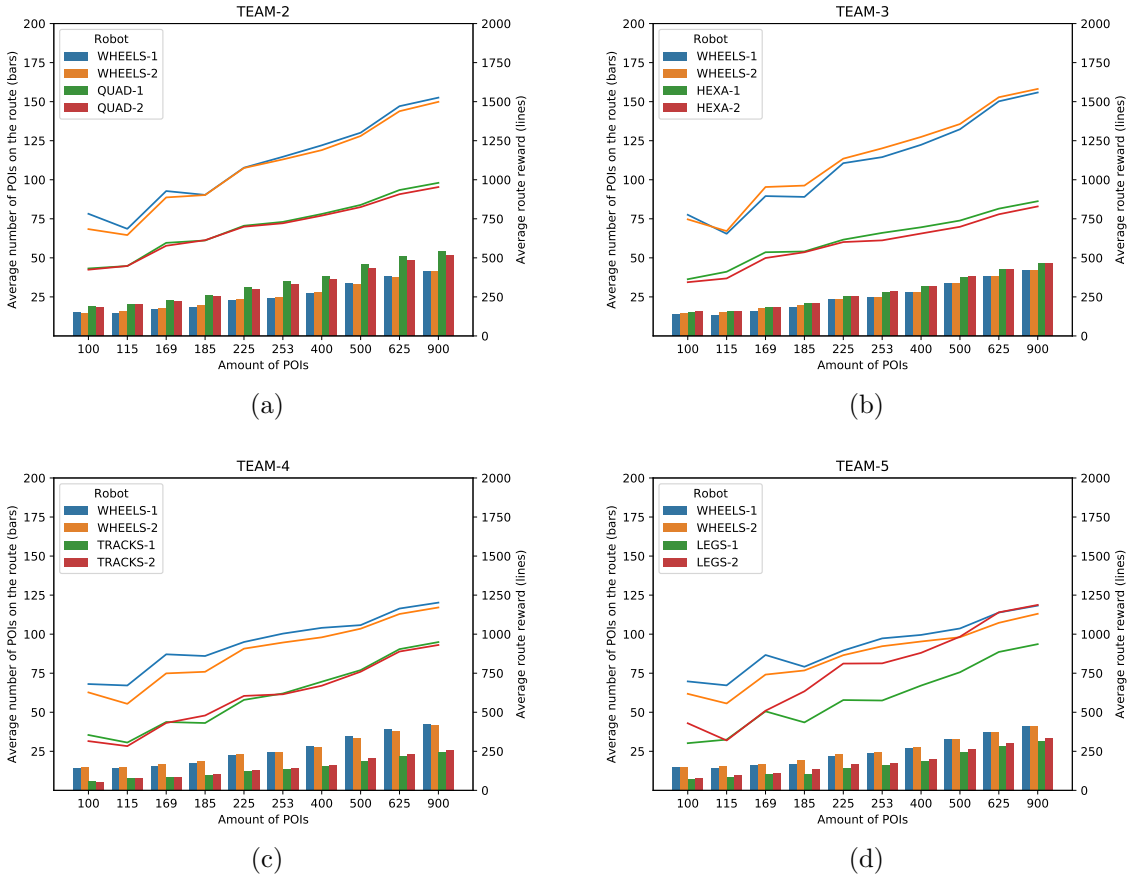


Figure B.13: Analysis of the routes generated for the picked heterogeneous teams. The average amount of POIs on each route is indicated with the bars (left axis), and the average reward of the routes is represented by the lines (right axis).

than SR had the lowest rewards. The sensing time also influenced the planned reward, causing teams with robots with shorter sensing times to collect more rewards than the others. The reward lost is similar for all teams except for the TEAM-1. Adapting the routes draws attention to the recovery of the TEAM-2, which is much greater than the others. The QUAD-type robots in this team are the fastest and have the shortest sensing times. For example, the HEXA-type robots of the TEAM-3 are as fast as the QUAD-type ones, but they take longer to sense a point. This delay reflects on reward recovery, as the TEAM-3 collects an equal or less reward than the homogeneous team.

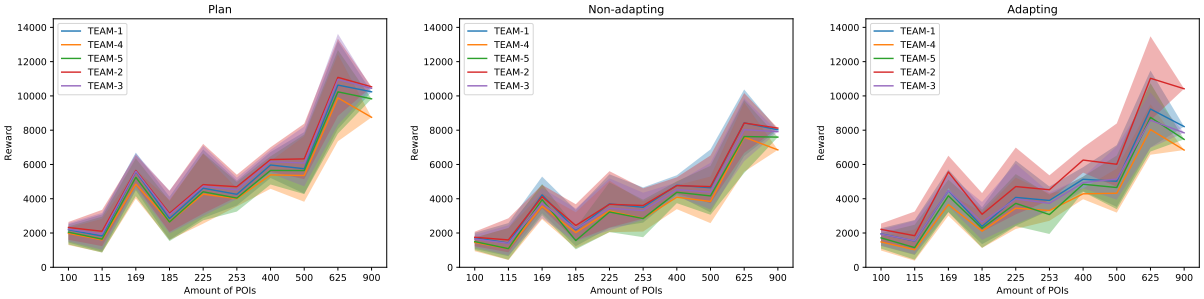
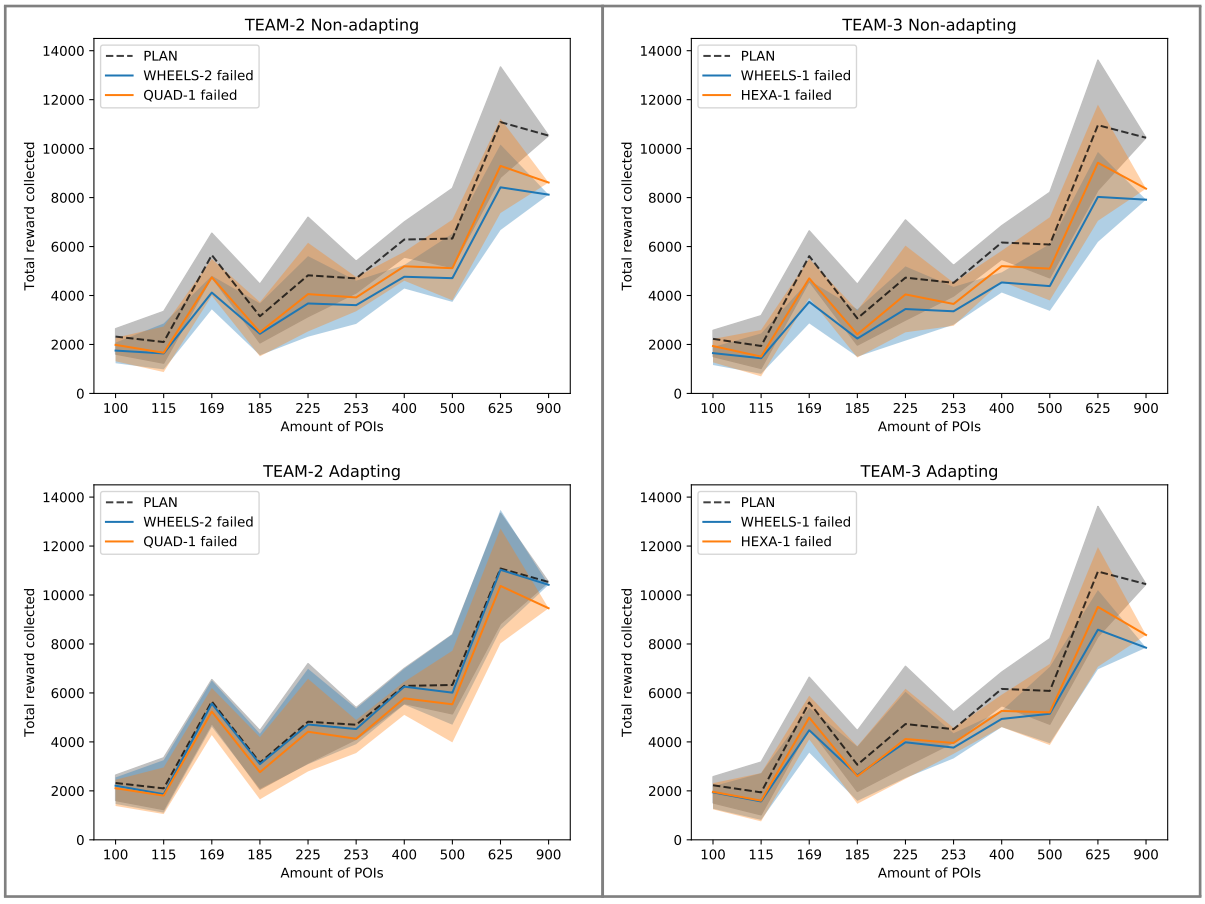


Figure B.14: Comparison of the computed average rewards considering the four spotlighted teams, according to the number of POIs in the instance.

Figure B.15, shows the cases where the pair of partner agents of the two SR are 50% faster than them. As WHEELS-type robots are slower in these cases, it is expected that the reward lost with the failure of these agents is greater, as they are responsible for the most profitable routes. The impact factor in these two results is the sensing times since agents of type HEXA are three times slower to sense a point. The situation that most call our attention in these results is the bottom plot of Figure B.15a, where we have the result of the mission after adapting the agents' route. When the WHEELS-2 failed, the reward recovery was very high, reaching almost all the reward lost with the failure. As we have seen, having agents with better resources favors the adaptation routine. The agents that have adapted their routes are particularly advantageous in this scenario, as they are faster and spend less time sensing the points than the failed agent.

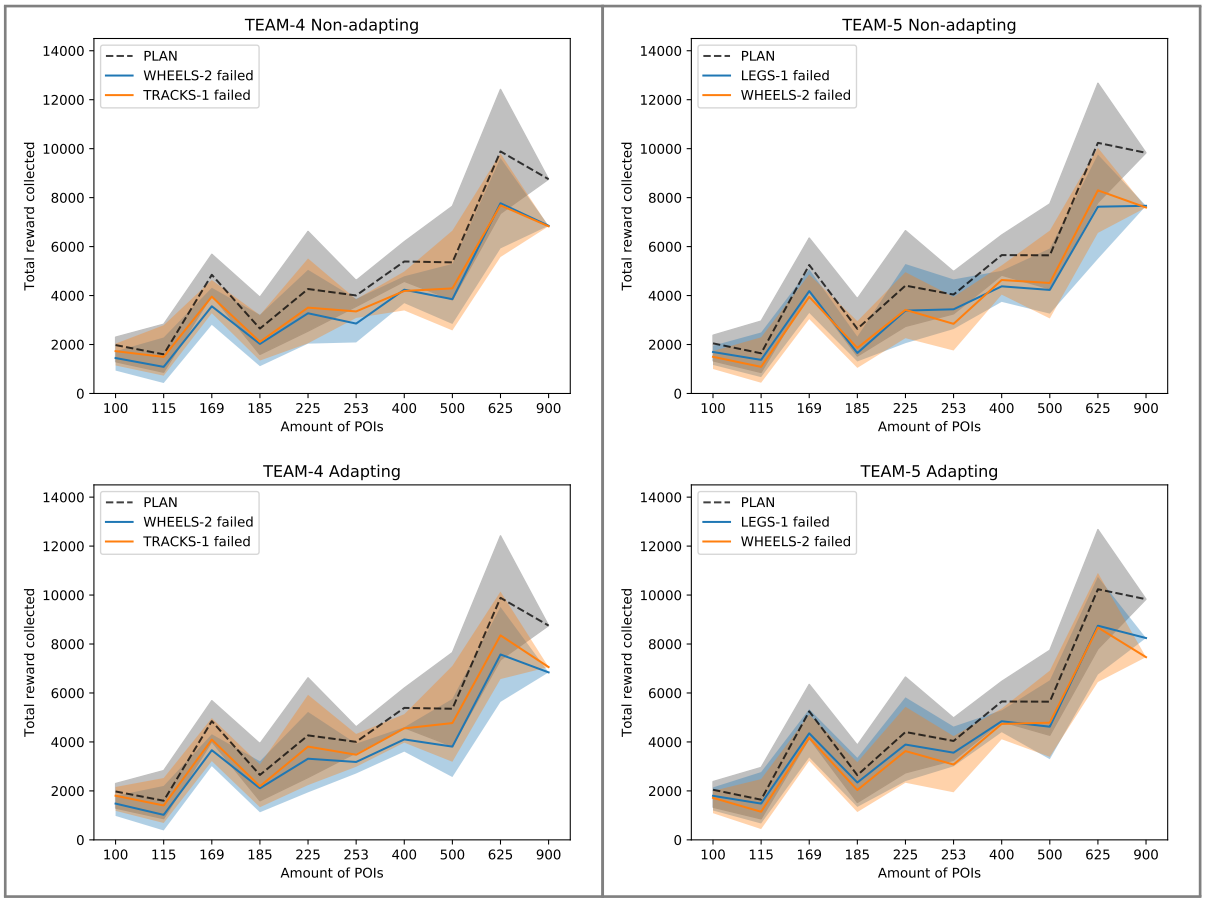


(a) (b)

Figure B.15: Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance. For each type of agent on the team, we force the failure of that responsible for the most profitable route. The upper plots show when the partners ignore the failure, and the lower when the agents adapt their routes. The black dashed line corresponds to the planned reward for the team.

Figure B.16 shows the cases where the pair of partners of the two SR are 50% slower than them. As these robots are very slow, the impact caused by their failures is

similar to when agents of type WHEELS fail. The sensing times also make a difference in the mission results when we use the route adaptation routine. TRACKS-type robots take twice as long to sense a point as a SR and 4 times longer than LEGS-type robots. This becomes clear when we look at the failure of WHEELS-2 in the lower plot of Figure B.16a. In this case, the reward recovered is much lower than when TRACKS-1 fails. This is because robots TRACKS-1 and TRACKS-2, in addition to being slower, take a long time to sense a point, which makes it difficult for them to inspect the points uncovered by the WHEELS-2. This does not happen when WHEELS-2 fails in TEAM-5. We see that even with the two slower LEGS-type robots in the team, the reward collected at the end was closer to when LEGS-1 failed. That's because LEGS-1 and LEGS-2 robots' sensing times are half what WHEELS-type agents spend, leaving more time to visit other points.



(a) (b)

Figure B.16: Comparison of the average rewards collected by the team when an agent of each type fails, according to the number of POIs in the instance. For each type of agent on the team, we force the failure of that responsible for the most profitable route. The upper plots show when the partners ignore the failure, and the lower when the agents adapt their routes. The black dashed line corresponds to the planned reward for the team.