UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA

Rodolfo Jordão

Planning for Cooperative Persistent Long Term Autonomous Missions

Planejamento para Missões Autônomas Persistentes Cooperativas de Longo Prazo

CAMPINAS
2018

Rodolfo Jordão

Planning for Cooperative Persistent Long Term Autonomous Missions

Planejamento para Missões Autônomas Persistentes Cooperativas de Longo Prazo

Master's thesis presented to the School of Mechanical Engineering of the University of Campinas, as one of the requirements for the Mechanical Engineering Masters program, in the area of Solid Mechanics and Mechanical Design.

Dissertação de Mestrado apresentada à Faculdade de Engenharia Mecânica da Universidade Estadual de Campinas como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Mecânica, na Área de Mecânica dos Sólidos e Projeto Mecânico.

Advisor: Prof. Dr. André Ricardo Fioravanti

ESTE EXEMPLAR CORRESPONDE À VERSÃO FINAL DA DISSERTAÇÃO DEFENDIDA PELO ALUNO RODOLFO JORDÃO, E ORIENTADO PELO PROF. DR. ANDRÉ RICARDO FIORAVANTI.

ASSINATURA DO ORIENTADOR

CAMPINAS
2018

Agência(s) de fomento e nº(s) de processo(s): CAPES, 1687532
ORCID: https://orcid.org/0000-0002-1277-390

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Luciana Pietrosanto Milla - CRB 8/8129

J767p

Jordão, Rodolfo, 1993-
Planning for cooperative persistent long term autonomous missions / Rodolfo Jordão. – Campinas, SP : [s.n.], 2018.

Orientador: Andre Ricardo Fioravanti.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Mecânica.

1. Planejamento. 2. Robótica. I. Fioravanti, Andre Ricardo, 1982-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Mecânica. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Planejamento para missões autônomas persistentes cooperativas de longo prazo
Palavras-chave em inglês:
Planning
Robotics
Área de concentração: Mecânica dos Sólidos e Projeto Mecânico
Titulação: Mestre em Engenharia Mecânica
Banca examinadora:
Andre Ricardo Fioravanti [Orientador]
Leonardo Tomazeli Duarte
Romis Ribeiro de Faissol Attux
Data de defesa: 26-02-2018
Programa de Pós-Graduação: Engenharia Mecânica

UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA MECÂNICA
COMISSÃO DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA
DEPARTAMENTO DE MECÂNICA COMPUTACIONAL

DISSERTAÇÃO DE MESTRADO ACADÊMICO

Planning for Cooperative Persistent Long Term Autonomous Missions

Planejamento para Missões Autônomas Persistentes Cooperativas de Longo Prazo

Autor: Rodolfo Jordão
Orientador: Prof. Dr. André Ricardo Fioravanti

A banca examinadora composta pelos membros abaixo aprovou esta tese:

Prof. Dr. André Ricardo Fioravanti
Faculdade de Engenharia Mecânica

Prof. Dr. Romis Ribeiro de Faissol Attux
Faculdade de Engenharia Elétrica e de Computação

Prof. Dr. Leonardo Tomazeli Duarte
Faculdade de Ciências Aplicadas

A Ata da defesa com as respectivas assinaturas dos membros encontra-se no processo de vida acadêmica do aluno.

Campinas, 26 de Feveiro de 2018

Acknowledgements

I'm very grateful to my advisors, parents, friends and close ones for their continuous support, and also to CAPES for financing my studies, which this works would not have existed otherwise.

Resumo

Uma metodologia para abordar missões autônomas persistentes a longo prazo é apresentada juntamente com uma formalização geral do problema em hipóteses simples. É derivada uma realização dessa metodologia que reduz o problema geral para subproblemas de construção de caminho e de otimização combinatória, que são tratados com heurísticas para a computação de solução viável. Quatro estudos de caso são propostos e resolvidos com esta metodologia, mostrando que é possível obter caminhos contínuos ótimos ou subótimos aceitáveis a partir de uma representação discreta e elucidando algumas propriedades de solução nesses diferentes cenários, construindo bases para futuras escolhas educadas entre o uso de métodos exatos ou heurísticos.

Palavras-chave: Planejamento, Roteamento, Design de Metodologia, Robótica Cooperativa

Abstract

A Methodology for tackling Persistent Long Term Autonomous Missions is presented along with a general formalization of the problem upon simple assumptions. A realization of this methodology is derived which reduces the overall problem to a path construction and a combinatorial optimization subproblems, which are treated themselves with heuristics for feasible solution computation. Four case studies are proposed and solved with this methodology, showing that it is possible to obtain optimal or acceptable suboptimal continuous paths from a discrete representation, and elucidating some solution properties in these different scenarios, building bases for future educated choices between use of exact methods over heuristics.

Keywords: Planning, Routing, Methodology Design, Cooperative Robotics

List of Figures

List of Algorithms

Glossary

ASC Admissible Score Clustering. 39, 42, 44, 45, 49, 50, 60, 63

AURORA Autonomous Unmanned Remote Monitoring Robotic Airship. 24

CTI Centro de Tecnologia de Informação Renator Archer.

CVRP Capacited Vehicle Routing Problem.

DRONI Conceptually Innovative Robotic Airship. 24

MILP Mixed Integer Linear Programming. 35, 37, 39, 42, 63

OOS Order and Orientation Switching. 42, 44, 49, 50, 60

PLTAM Persistent Long Term Autonomous Mission. 23–26, 29, 35

ReSC Restarting Sequential Construction. 42, 49, 50, 60

ShorPaP Shortest Path Problem. 30, 35, 36, 63

SimPaD Simple Path Drift. 30, 32, 63

TSP Traveling Salesman Problem. 39, 42

UAV Unmanned Air Vehicle. 25, 49, 50

UGV Unmanned Ground Vehicle. 34, 49, 50

VRP Vehicle Routing Problem. 29, 30, 39

Symbols

E^{c+} Cell graph edges, being primitives.

E^{d+} Decision graph edges, being primitives.

G^c Cell graph, resulting from segmentation.

G^d Decision graph, resulting from path construction.

I^a Index of agent classes.

I^o Index of operations.

I^r Index of interest regions.

N^{c+} Cell graph nodes, being positions and orientations.

N^{d+} Decision graph nodes, being positions and orientations.

R_i Sub interest region of index i.

R Working total region.

β_a Capacity limits for agent class a.

$\mathbb{N}$ Set containing all natural numbers (without zero).

$\mathbb{R}$ Set containing all real numbers.

$\mathcal{P}(b^R)$ Set of discrete paths that originate and end in b^R.

$\mathcal{P}^C(b^R)$ Set of discrete paths that originate and end in any other point than b^R.

$\mathcal{S}(b^R)$ Set of continous paths that originate and end in b^R.

$\mathcal{S}(p_1, p_2)$ Set of continous paths that originate in p_1 and end in p_2.

$\mathcal{T}$ Set of graph transformations.

μ_a^c Cost functional for agent class a.

μ_a^e Efficiency functional for agent class a.

$\mu_{a,o}^e$ Supply functional for agent class a and operation o.

ρ^c Maximum covering radius of all agent classes.

Φ Set of discrete orientations.

b^R Base location.

r_{min} Minimum turning radius of all agent classes.

Contents

1 INTRODUCTION

This chapter discusses the motivations and related previous research present in the literature. Therefore, if the reader is already familiar with the matter of this text, this chapter can be safely skipped, with the exception of the platform overview at Section 1.4.

Persistent Long Term Autonomous Missions (PLTAMs) refer to autonomous operations which are designed to last long periods of time with the least external intervention as possible. Examples include autonomous agricultural harvesting or continuous surveillance in urban or wild environments (TSOURDOS; WHITE; SHANMUGAVEL, 2010), such as the development of cooperative aerial robots for environmental monitoring in large forest regions like the Brazilian Amazon (PINAGÉ; CAVALHO; QUEIROZ-NETO, 2013). Figure 1.1 shows a sampling and surveillance mission with a central base that the agents, or robots, return in order to refuel and exchange information.

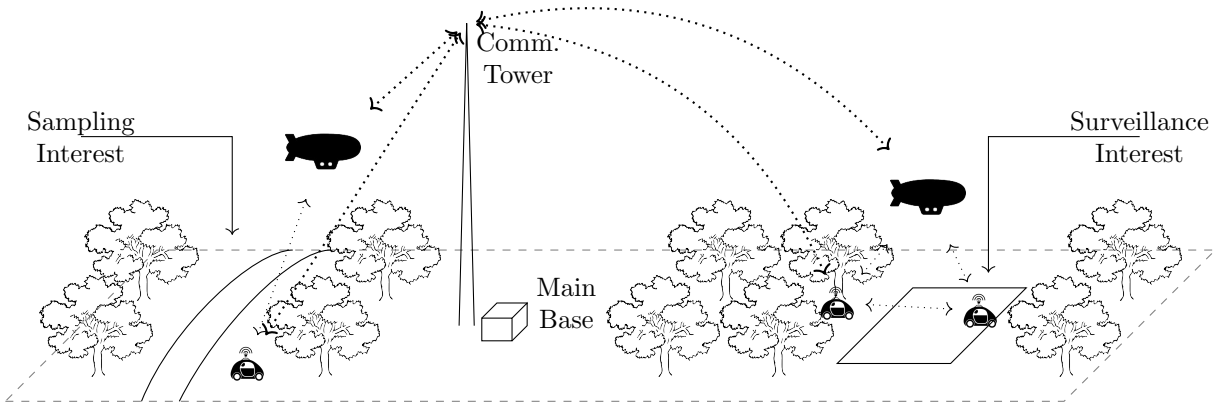

Figure 1.1: Example of a Persistent Long Term Mission with sampling and surveillance objectives.

A minor problem regarding PLTAMs, or Autonomous missions in general, is nomenclature. There is not, as of 2018, naming conventions in the literature for problems similar to the one treated in this thesis; the closest ones are called robotic coverage problems and may have very different constraints with similar goals. Section 1.3 lists some previous relevant work that may employ other names for their challenges. Here, the term persistent autonomous mission or persistent autonomous operation will be used for multi-agent systems whose goal is executing some tasks optimally, whatever the metric, in a repeatable manner, while respecting all agents running constraints, e.g. fuel.

1.1 WORK INSPIRATIONS

Persistent Long Term Autonomous Mission (PLTAM) is not a new concept in engineering practice. For instance, platforms destined to exploration have already for many years now embraced this paradigm as the safest and most economical approach, the most famous of these late platforms being the discovery autonomous Mars rover (NASA, 2017). Other cases where autonomous missions can be classified as PLTAMs occur in military operations of surveillance, agriculture in automated farming and, more recently, safe commercial self-driving connected vehicles. All these problems have similar descriptions in which a metric, usually time or operation cost, must be minimized by the cooperation of all agents simultaneously, while respecting their constraints and ending in the same place they started: a “base”, thus, conforming to the definition of PLTAM given earlier.

In Brazil, a continued research effort aims to study an autonomous surveillance systems for the Amazon rainforest, in order to contain illegal deforestation that has been decreasing in the last years but is still significant to greatly impact the ecosystem as a whole. Figure 1.2 shows the deforestation estimates for the legal Amazon region in Brazil from 1988 to 2017 (INPE, 2017).

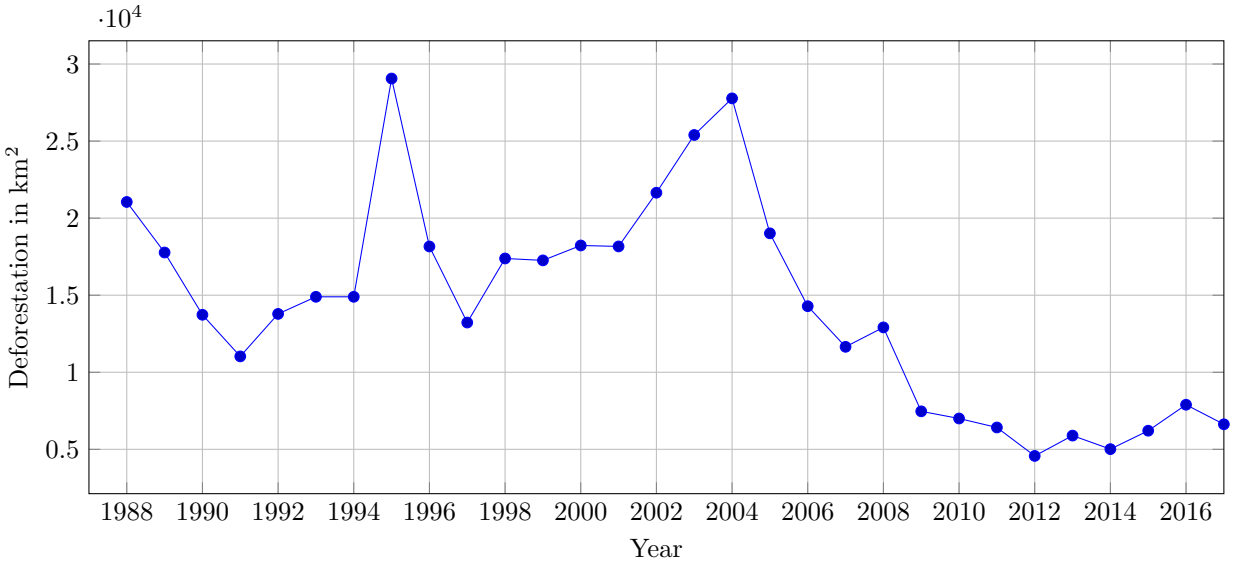

Figure 1.2: Deforestation of Brazil's Amazon legal region, from 1988 to 2016.

Two projects that are part of these efforts are the robotics airships Autonomous Unmanned Remote Monitoring Robotic Airship (AURORA) (ELFES, ALBERTO et al., 1998) and Conceptually Innovative Robotic Airship (DRONI), the latter being a reworking and improvement of the former, destined to monitor large areas of the Amazon with the least supervision possible, i.e. in a PLTAM. Figure 1.3 shows an AURORA implementation in test and a DRONI geometric model concept.

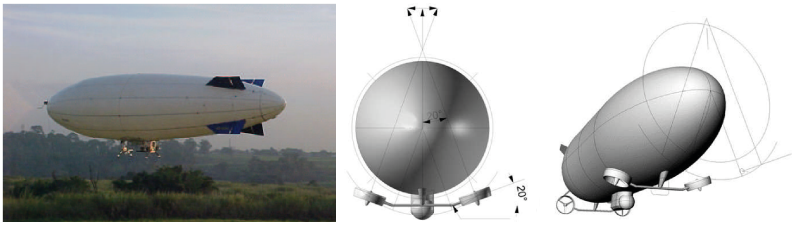

Figure 1.3: The AURORA (left) and DRONI (right) projects.

Once again, all these projects share the common assumption that the robots must execute tasks cooperatively in a certain map and must return to a prescribed point, a base, so that they refuel and exchange information. This abstraction is the key inspiration for all work done in this thesis, as will become apparent in Chapter 2. Hence, although no explicit real-life application is studied in this text, the eventual implementation of a planning methodology for these autonomous amazon surveillance missions constitute the goal that the theoretical exploratory studies in this text have.

1.2 OBJECTIVES

The main objective of this thesis is to build a framework of techniques which permits the planning of cooperative persistent long-term autonomous missions with the minimum overall mission cost on a long-term basis.

1.3 RELATED WORK

As mentioned earlier, a similar problem to the PLTAM is repeated robotic area coverage, sometimes also treated as one-off area coverage. In these problems, the work done by Fazli, Davoodi, and Mackworth (2013) considers a group of homogeneous robots for repeated area coverage, starting from a region description for robotic task construction, although non-holomonic constraints are not naturally embedded into the solution workflow. Closer to this one, the research developed in Xu, Viriyasuthee, and Rekleitis (2014) tries to segment and route the surveillance in a region to homogeneous Unmanned Air Vehicles (UAVs) agents, treating non-holomonic agents via after-route greedy techniques. The major difference is that the methodology here considers the agents constraints directly during planning, via creation of "state graphs", somewhat similar to the idea developed by Pivtoraiko, Knepper, and Kelly (2009), but treating the objectives in their natural domain, rather than the agents'.

Some coverage problems are treated three dimensionally to better fit the operation in question, for instance, in Bentz et al. (2017), a workflow was developed for inspection missions where multiple homogeneous UAVs cover a 3D map with defined visual sensors and resampling. In Bircher et al. (2016), a similar objective was sought, but the energy capacity of the UAVs was considered along with the coverage of the 3D environment, resulting in the return of the covering agents to a refuel station near their end of energy capacity. Although 3D approaches are able to model a greater number of robotic missions, there is still a major drawback allied with the complexity of choosing to do so: the works so far usually model the problem as a control one, increasing the difficulty of optimality provability when compared to modeling problems in a pure optimization fashion, as done here. Note that this observation is also valid for 2D ambiances, independently of the reduced dimensional complexity.

There is also the study and design of a completely decentralized schemes: in Javanmard Alitappeh et al. (2017), agents are routed in a general topological map in way suited for indoor missions, although details of its construction were not the major focus; in Mitchell et al. (2015), a decentralized scheme for routing that respects the capacity and refueling constraints of the robots was developed and tested in real robots, planning a PLTAM successfully for about 10 robots. Aligned closely to off-line planning, in Broderick, Tilbury, and Atkins (2014), a topological deconstruction in a sweep manner is done in a bidimensional map, then treated as optimal control problem where the covering mechanic happens due to the cost function. Other similar theoretical works include also an optimal planning scheme for choosing the best swiping order, where swiping is the linear geometry the agents make to cover the environment, cooperatively in an agricultural setting with many agents (BURGER; HUISKAMP; KEVICZKY, 2013) and a continuous area coverage discretized by methodical sampling of the domain, followed by routing and assignment of a single UAV (ISAACS; HESPANHA, 2013). The main difference of the methodology proposed here is that it does not require a priori setup of swiping motion geometries commonly found in general coverage missions while also treating the continuous planning in a discretized manner.

Other studies include treatment of these persistent missions in a more concrete fashion, researching and developing a complete Unmanned Aerial Systems with similar considerations as developed here, but focusing on implementation, homogeneous agents, testing and system deployment aspects (DAMILANO et al., 2013; BOCCALATTE et al., 2013), in contrast to the problem abstraction and solution focus of this work. That is, their main goal is the hardware and software required for implementation of a system such as the one displayed in Figure 1.4.

Additionally, the heuristic family proposed in this text takes inspiration from tabu-search heuristics and the seminar clustering heuristic proposed by Fisher and Jaikumar (1981), but relies equally on both graph structures and optimization model, unlike other approaches that tend to treat the optimization model directly (CAMPOS; MOTA, 2000). Moreover, the algorithms proposed here do not require a startup procedure, e.g. seed selection, although this may be studied in the future to increase the heuristics' efficiency.

1.4 PLATFORM DESIGN

A methodology can be proposed here, tuned to PLTAMs missions, that consists in a planner knowing a model for each agent class and using this model to estimate operational costs while this agent is performing

a task in route, then, with these estimates, the agents can be ordered and assigned accordingly. As a consequence, optimality, as defined per mission, could be achieved via exact methods to the accuracy of these estimations.

In certain sense, this methodology tends to be centralized, as agents try to follow the planner orders as close as possible, but also gives each of them relative autonomy to preserve themselves by disobeying orders if necessary. An implementation of it is shown diagrammatically in Figure 1.4, where some information flows are named for easier visualization. Note that the biggest autonomous aspects refer to the methodology and platform as a whole, rather than the planner or the agents alone, since it can be rightfully argued that this platform transfers the autonomy of the agents to the planner.

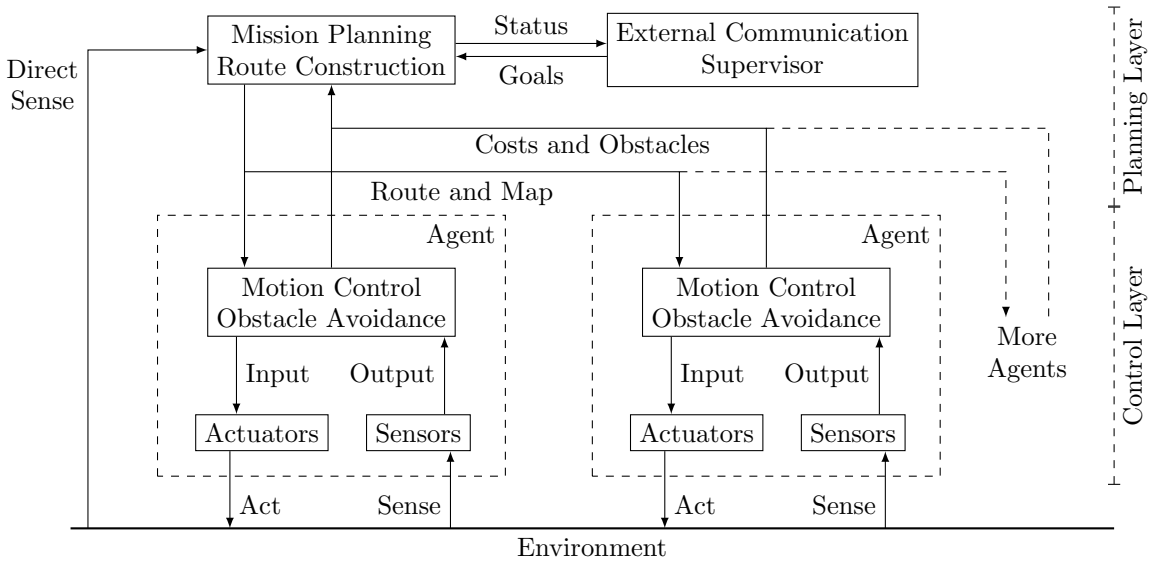

Figure 1.4: Overall autonomous platform view and its inter-relations.

Using Figure 1.3 as a visual aid, it is clear that the platform possesses a hierarchical structure, with the planner functioning as a "hive mind" and issuing the believed best actions for all agents, based on its knowledge of them and the environment. This knowledge can optionally be updated with every round-trip the agents make, based on their collected data, and on some instrumentation that the planner may have itself, an idea that is not pursued in this initial study, but is left as a possible follow-up with in Chapter 4. Once issued and order, or used here as synonymous, a task, a route or an tour, it must strive to maintain all orders given by the planner whilst being able to alter some actions depending on the environment for self-preservation. For example, it can stray from its given route by circumventing static obstacles in the environment or dynamic obstacles, like another agent. Therefore, it is plausible to affirm that this architecture fits well missions where the target region is relatively known a priori and has mild temporal variations, as are the majority of the inspirational missions here.

1.5 MAIN CONTRIBUTIONS

This work main contributions are twofold:

1. a formalization of the PLTAM problem in relatively general form, assuming that the only refuel station is the base itself and that the agent must always return to it, in Chapter 2,

2. development of a solvable realization for said formalization with an accompanying solution workflow that can treat heterogeneous and non-holomonic agents, in Chapter 2, and its validation via a number of case studies, in Chapter 3.

The results and conclusions were also summarized into a journal paper entitled "Cooperative Planning Methodology for Persistent Long Term Autonomous Missions" which is currently under review.

2 MISSION PLANNING

This chapter deals with cooperative PLTAMs planning for the platform, as to maximize the agents efficiency in their missions.

2.1 PROBLEM FORMALIZATION

Consider that there is a region $R \subset \mathbb{R}^2$, subregions of interest $R_i \subset R$ indexed by I^r and a point $b^R \in R$, acting as source of all agents. Let I^a be the index set of all agents classes, not their members. Let $\mathcal{S}(b^R)$ be the set of all cyclic directional paths in $\mathbb{R}^2$ adjoined with their path-following orientation, that start and end in b^R; and let I^o be the index set of all operations possibly performed by all agents.

Also, define indexed functionals, μ^e_a, $\mu^d_{a,o}$ and μ^c_a, so that $\mu^e_a : \mathcal{S}(b^R) \to \mathbb{R}$ gives every path s a cost measure, such as the amount of energy spent moving and performing operations in s, by the agent of class a; $\mu^d_{a,o} : \mathcal{S}(b^R) \times R \to \mathbb{R}$, represents the amount that the path s supplies of operation o to the subregion R_i, if performed by agent class a; and $\mu^c_a : \mathcal{S}(b^R) \to \mathbb{R}$ maps every path s to a capacity measure, which may coincide with μ^e_a, for every agent class a. In these lines, let $\alpha_{i,o} \in \mathbb{R}$ represent the minimum amount of demand that a region $R_i \subset R$ requires of operation o, and let $\beta_a \in \mathbb{R}$ represent the maximum amount of capacity an agent is able to effect.

The planning model can then be stated as an optimization process by simultaneous construction and assignment of variable number n of routes $s_1, \ldots, s_n$ to the agent classes,

$$\min_{\substack{n \in \mathbb{N} \\ s_1,\ldots,s_n \in \mathcal{S}(b^R)}} \left(\sum_{i \in \{1,\ldots,n\}} \min_{a \in I^a} \mu^e_a(s_i) \right), \tag{2.1}$$

as long as the assigned agents, $a_i = \operatorname{argmin}_{a \in I^a} \mu^e_a(s_i)$, meet all subregions demands,

$$\sum_{i \in \{1,\ldots,n\}} \mu^d_{a_i,o}(s_i, R_j) > \alpha_{j,o}, \quad \forall j \in I^r; o \in I^o, \tag{2.2}$$

and have their capacity limits respected,

$$\mu^c_{a_i}(s_i) < \beta_{a_i} \quad \forall i \in \{1, ..., n\}. \tag{2.3}$$

Equations (2.1)-(2.3) from now on consists of the true problem to be solved. This generalization is not a unification in planning and optimization, as previous works in the literature (VIDAL et al., 2014, 2013) settled for Vehicle Routing Problems (VRPs) in general, which is one of the core subjects for this work; rather, it is a fairly general mathematical abstraction tied to continuous planning and routing, in essence, abstracting the concepts of goal, demand and limits. For example, the cost functional could be defined upon energy, where as the capacity functional could be defined upon trajectory time estimation, but for all purposes of this work, the cost and capacity functionals coincide, being the operational cost of a given agent class: $\mu_a(s) = \mu^c_a(s) = \mu^e_a(s)$.

Due to the abstractness of this optimization model, restatements are necessary in order to solve it. One approach is to separate the problem into construction and assignment, whereas the second stage works upon the built paths of the first. Once this separation is made, the entire region R can be segmented so that both planning subproblems are treated in a discrete fashion and can be solved efficiently via suitable methods. This complete discretization has the added benefit that both generated subproblems are very similar or equal to two other widely studied problems in the literature, namely, Shortest Path Problems (ShorPaPs) and VRPs, to give weight to the efficiency claim. This workflow for discretization followed by construction and assignment is shown in Figure 2.1, where dashed blocks represent names of stage results while continuous ones represent stages.

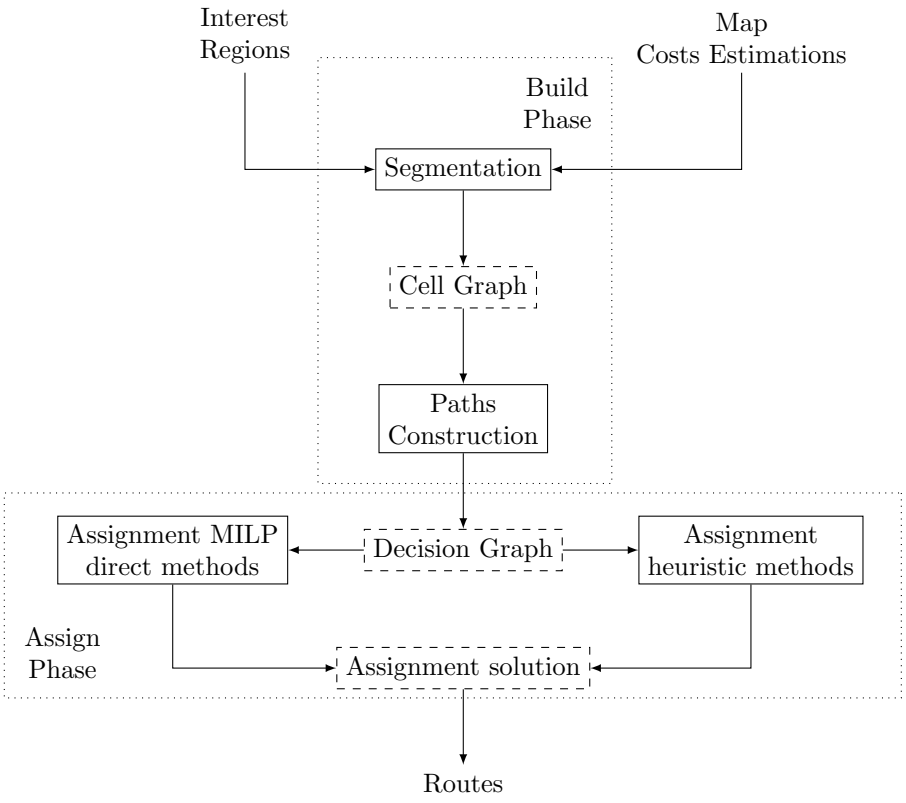

Figure 2.1: Mission planning workflow diagram.

Every stage shown in Figure 2.1 may also have substages associated with their smaller specific prerequisites. Segmentation, as will be expanded in Section 2.2, must address the orientation of the agents and the inherent segmentation error. The path construction, expanded in Section 2.4, gets the resulting segmentation discrete representation as a digraph and thus must be able to connect every possible state segmentation using only this graph, and finally, the assignment substage, expanded in Section 2.5, must be able to build tours for all agent classes knowing only the strongly connect graph resulting from these previous substages.

Also, note that grouping together agents into classes is in fact beneficial, for if there are less agents than routes, these will simply be cycled at the availability of said agents, and more importantly, if there are more agents than routes, the route assignment can cycle every individual, distributing concentrated effort stress and granting higher robustness against unforeseen problems during operation.

2.2 SEGMENTATION

The segmentation scheme chosen is required to fulfill three requirements. First, conserve the original cost, demand and covering aspects of the original continuous problem. Second, and minimize inherent segmentation error between the best arbitrary continuous path and the best discretized path, the Simple Path Drift (SimPaD), which is illustrated with examples in Figure 2.2, where the difference between the true best path becomes more pronounced as the lattice is made coarser.

Third, any segmentation scheme must be aware of orientation during the path making process to contemplate agents that are non-holomonic and more precisely estimate the cost of operation by any of the agents, otherwise, if this account is not made at the segmentation substage, the generated paths cost estimates will become hard to predict, or even impossible to follow by non-holomonic agents, beating one of the advantages of ahead-of-time planning central to this methodology. Figure 2.3 shows the difference of a continuous path that would be rebuilt solely looking on the previous traversed node of each arc visited and another rebuilt by planning the orientations of each node along their positions, shown in

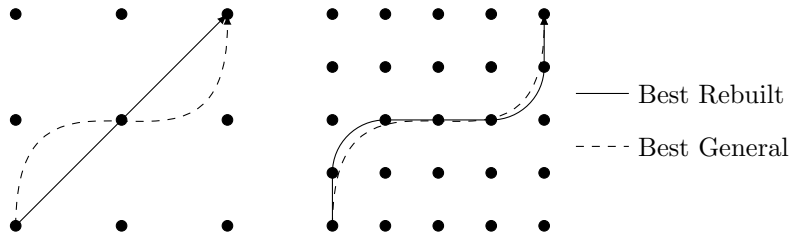

Figure 2.2: Two examples to highlight the segmentation effect on the drift from the best true path obtained after segmentation.

solid lines, for the same path without considering orientations, shown in dashed lines: not only the first reconstruction is more distorted, but it would also be harder to determine a more precise estimate of the cost necessary to traverse it, since the continuous reconstruction must happen after all positions are set.

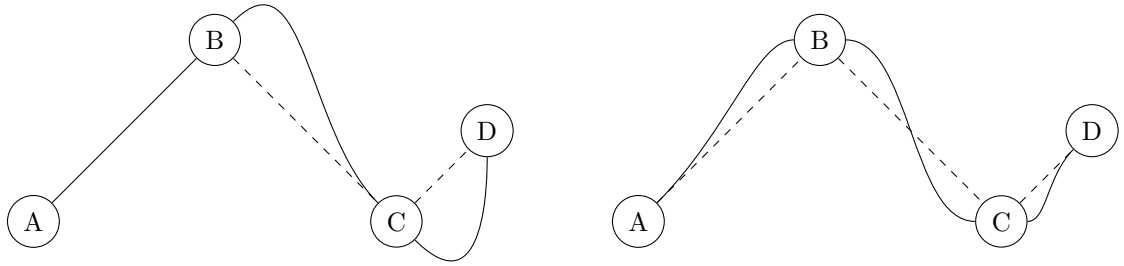

Figure 2.3: Two examples of reconstruction after position planning (left) and during position planning (right) highlighting the necessity of treating orientation with positioning.

Lastly, although not strictly required, it is desirable that the scheme be simple enough so that modest hardware are able to solve all algorithms in soft real-time, so that it can more easily, and cheaply, be deployed anywhere.

One scheme fulfilling these prerequisites is a quadrangular, uniformly spaced, grid as shown in Figure 2.4, where the lattice gap needs to be small enough so that no patch of region is left uncovered, but also must be big enough so that the agents are able to follow the planner generated paths. In other words, the gap, denoted by l, must lie between the agents' minimum turning radius r_{min} and their covering radius, ρ^c, the maximum possible value of this gap being $\sqrt{2}\rho^c$.

Additionally, for every point a finite number of continuous paths connecting other neighboring points are created, called here primitives and exemplified visually in Figure 2.4, so that orientation can be treated along positioning as mentioned earlier, effectively creating a state lattice for all agents, thus giving nodes an orientation and a position based on these primitives. Here, in accordance with the square-like segmentation, eight angles are chosen as equal divisions of the trigonometric circle.

This scheme is part of a broader class of segmentations called grid-based methods, since they involve repeated patterns that represent the original continuous domain. A current popular alternative to grid-based methods are those based on sampling, termed sampling-based methods, which can treat more complex cases more easily as they involve literally repeated operational domain exploring and sampling until some criteria is met. This means they can depend solely on the mathematical model of an agent class, but also, this means that contrary to grid-based methods, sampling ones may not induce lattices (PADEN et al., 2016). This weak regularity, while very useful in other scenarios, is detrimental for the problem treated here, as it is necessary to always check that no single point of the plane is left uncovered and keep sampling in these regions accordingly. Still, repetitive sampling on regions that are uncovered may create unintentional dense clusters of points in some areas to cover R completely. Therefore, without special modifications, grid based methods are more attractive for this covering segmentation than sampling based ones, despite their better exploratory and computational performance in more general

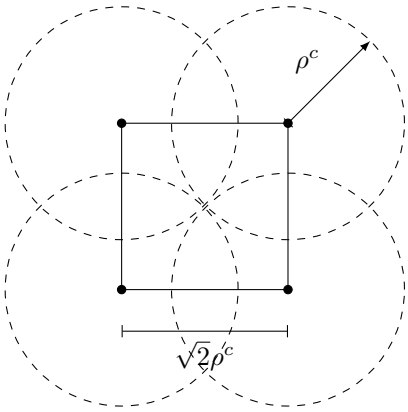

Figure 2.4: Visual representation of the lattice used in the segmentation. In this case showing the maximum possible lattice gap.

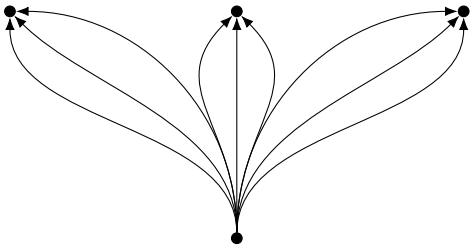

Figure 2.5: Example of primitive and neighborhood definitions.

and complex scenarios (KARAMAN; FRAZZOLI, 2011). Further enhancements and alternatives to this simple quadrangular scheme are discussed in Chapter 4.

An advantage of this state lattice inducing scheme is that the state neighborhood is defined on a handful of continuous primitives, for a single state or node, which also means that complex continuous maneuvers can be described by the sum of simpler constituent primitives as seen in Figures 2.6. Note that these reconstructions must allow non-holomonic agents to follow it, hence the primitive function classes, the lattice gap and the neighboring definition collectively play a significant role in the agents' path tracking success.

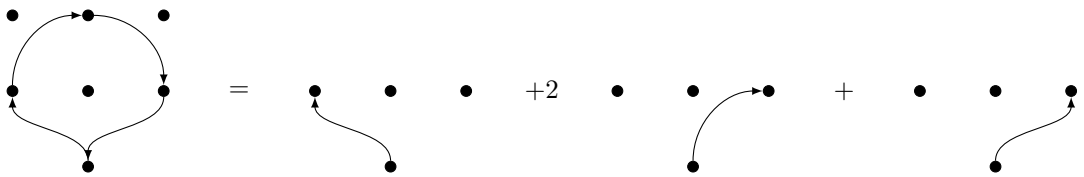

Figure 2.6: Constituent simpler elements of a complex maneuver.

The last concern for the success of this segmentation scheme, aside from choosing the primitives and node gap, is the positioning of the grid itself over the region, as it can "glide" and rotate above it. For simplicity, it is possible to assume that rotations are unnecessary as the regions is already in a rotated frame that fits the quadrangular segmentation very well, either by some pre-processing or hand-crafting, leaving out translation to be addressed. Naturally, it is desirable to minimize the SimPaD and get the best discretization; this, however, implies that the continuous paths are known beforehand, for all variations of gaps, primitives and dislocations chosen, or at least some SimPaD estimate that depends on these features is known. An alternative is to maximize the number of points contained in the region, guided by the thought that the SimPaD gets smaller by increasing the amount of reconstruction possibilities available. Therefore, the segmentation dislocation can be made by finding the displacement of highest point count inside the region, or, since major importance are given to the interest subregions, the grid can be biased towards interest locations by recounting points inside them. Note that the points are not

the nodes themselves, but points with a orientation attached.

To effect this point maximization for a planar region R, let $\delta_R(p)$ be an indicator function that returns 1 if the point $p \in \mathbb{R}^2$ is in region R and 0 otherwise,

$$\delta_R(p) = \begin{cases} 1, & p \in R \\ 0, & \text{else} \end{cases}, \tag{2.4}$$

and let $d \in \mathbb{R}^2$ be the displacement vector in the plane that contains R. Consider all the points that are generated by the segmentation scheme in r indexed by the set i^p. Then, the positioning maximization can be stated as,

$$\max_{d \in \diagdown^2} \sum_{p \in i^p} \delta_r \left(d + p\right). \tag{2.5}$$

If the bias is to be used, let λ be a positive real number, and δ_{R_i} a pertinence function for each of the interest subregions R_i, in the same fashion as δ_R. Then the optimization problem can be restated as

$$\max_{d \in \mathbb{R}^2} \sum_{p \in I^p} \left(\delta_R \left(d + p\right) + \lambda \sum_{i \in I^r} \delta_{R_i} \left(d + p\right) \right), \tag{2.6}$$

where solutions that do not contain at least one point inside every subregion R_i are discarded,

$$s.t. \quad \sum_{p \in I^p} \delta_{R_i} \left(d + p\right) > 0, \quad \forall i \in I^r \tag{2.7}$$

Solving (2.6)-(2.7) efficiently requires knowledge of limits for d. Fortunately, these bounds can be found when the quadrangular symmetry of the packaging scheme is considered, so that whenever the mesh is moved more than one lattice gap, the same exact mesh is obtained, as exemplified visually by Figure 2.7, where the region R is circular.

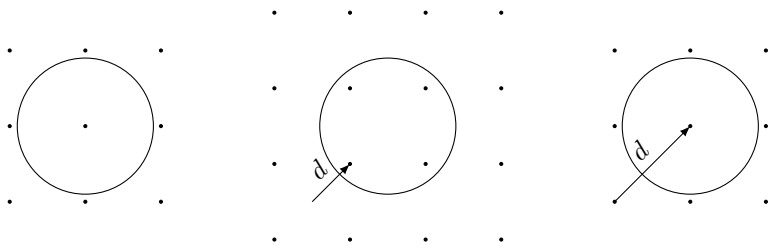

Figure 2.7: Example of equivalent lattices achieved via displacement. The left lattice is the original, the middle lattice is the same as the left one, but displaced diagonally, and the right one is the lattice obtained by displacing a full diagonal from the left one and half diagonal from the middle one.

Therefore, it is enough to have both displacements be contained in the interval of 0 to the lattice gap, given by l, which is shown in Figure 2.8 by the new position of point A, A', generated by the displacement d. The final displacement optimization model is thus given by

$$\max_{d \in [0,l]^2} \sum_{p \in I^p} \left(\delta_R \left(d + p\right) + \lambda \sum_{i \in I^r} \delta_{R_i} \left(d + p\right) \right), \tag{2.8}$$

subject to restriction (2.7).

The segmentation results in two entities: the position points themselves in a set N^c and a graph $G^c = (N^{c+}, E^{c+})$ named state cell graph, in which the nodes, as defined earlier, are the positions in N^c augmented with a finite set of orientations $\varphi \in \Phi$, resulting in the node set N^{c+} and its connections E^{c+}. Further possibilities of segmentation are discussed in Chapter 4. Next, the cost estimation of every edge in E^{c+} is discussed.

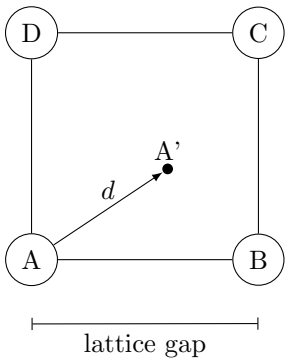

Figure 2.8: Intervals of search for positioning.

2.3 COST ESTIMATION

To be able to estimate costs between the generated nodes in the segmentation, a representative model of each agent class is required so prediction of the amount of effort spent between two desired nodes can be made. As such, some aspects of control and dynamics need to be discussed with their relations to the discretization.

Alluding established nomenclature from the literature (LAVALLE, 2006; SICILIANO, 2009), there are mainly two concerning spaces for any description, the operation or trajectory space $\mathcal{S}$, where, as the name implies, all possible trajectories and operations happen, and the configuration space $\mathcal{X}$, where all general information of the agent, such as joint positions, engine thrusts, etc lies. The former, in this work, is given by $R \times [0; 2\pi]$, since all trajectories are described by a history of plane positions and an angle of orientation; while the latter can change wildly from agent to agent.

To make these last ideas more formal, consider that an agent a has generalized coordinates $q \in \mathcal{X}_a \subset \mathbb{R}^n$ that completely describe it, a number of free variables $u \in \mathcal{U}_a \subset \mathbb{R}^m$ that works as the input in the system, and finally a model $f_a : \mathbb{R} \times \mathcal{X}_a \times \mathcal{U}_a \to \mathcal{X}_a$ that represents the agent,

$$\dot{q}(t) = f_a(t, q(t), u(t)), \tag{2.9}$$

where $t \in \mathbb{R}$ symbolizes "time" in a context that makes sense for this description. Equation (2.9) is the general form of any agent description in the configuration space, however, the objective is to control the agent trajectory rather than its configuration, i.e. controlling in $\mathcal{S}$ rather than in $\mathcal{X}_a$. Thus, every f_a must be coupled with a transformation, or output, function $h_a : \mathbb{R} \times \mathcal{X}_a \to \mathcal{S}$,

$$s(t) = h_a(t, q(t)), \tag{2.10}$$

where $s(t) \in \mathcal{S}$ for any $t \in \mathbb{R}$. Finally, there may be constraints on the agent that forbid it to make impossible actions, such as a Unmanned Ground Vehicle (UGV) instantly making a half-turn with zero radius to backtrack somewhere; represented by the function $g_a : \mathbb{R} \times \mathcal{X}_a \times \mathcal{U}_a \to \mathbb{R}$,

$$0 = g_a(t, q(t), u(t)). \tag{2.11}$$

Equations (2.9) to (2.11) form the basis of any agent description, where the control objective is to make $s(t)$ follow a predetermined reference path $s^r(t)$. It should be remarked that the symbol $\mathcal{S}$ was used here intentionally, to reinforce its relationship with $\mathcal{S}(b^R)$ from the overall formalization, seeing that every s^r is, in fact, a member of $\mathcal{S}(b^R)$ and that $R \subset \mathcal{S}$: the symbolism of applying $\mathcal{S}$ to a point should be understood as a "filter" function that returns only the continuous paths that start and end at that point, in the same sense, $\mathcal{S}$ applied to two different points, $\mathcal{S}(p_1, p_2)$, should be understood as the set of all continuous paths that start at the first point and end at the second.

Using these definitions for costs estimation, consider a path $s \in \mathcal{S}(p_i, p_j)$, for any $p_i \in \mathcal{S}$ and $p_j \in \mathcal{S}$, a minimum speed v_a^{min} and generalized coordinates $q_a \in \mathcal{X}_a$ so that in a time span $t \in [0; T]$, $s(t) = h_a(t, q_a(t))$, that is, the agent follows perfectly s in $[0; T]$ by means of a control $u_a(t)$ while respecting its

constraints,

$$\begin{aligned} \dot{q}_a(t) &= f_a(t, q_a(t), u_a(t)), \\ 0 &= g_a(t, q_a(t), u_a(t)), \\ |\dot{s}(t)| &> v_a^{min}, \\ s(t) &= h_a(t, q_a(t)), \end{aligned} \tag{2.12}$$

the cost function of s, $\mu_a(s)$, is defined as

$$\mu_a(s) = \int_0^T L_a(u_a(t))dt, \tag{2.13}$$

where L_a is a strictly positive function that takes the control vector u_a and outputs a scalar number, for each agent class a, representing its cost metric. The minimum speed is a simple mathematical device to incorporate the possibilities of the agent using the environment in its favor when possible, e.g. a UGV going down a hill may let its speed increase instead of stopping it. Naturally, unless the agent has perfect energy savings, or spontaneous energy generation, the cost function is strictly positive for any non-zero cyclic path s starting and ending at any point with minimum speed v_a^{min},

$$\mu_a(s) > 0, \quad \forall a \in I^a.$$

Back to segmentation, this continuous cost must be described in terms of nodes and primitives. Consider a path P formed by two neighbor nodes in G^c, (i, φ_i) and (j, φ_j),

$$P = \{(i, \varphi_i), (j, \varphi_j)\},$$

and let s be the path formed by the primitive connection between these two nodes as chosen at the segmentation. The cost of P is then defined to be the cost of s,

$$\mu_a(P) = \mu_a(s), \tag{2.14}$$

and in the same line, the cost variable representing this edge, that is, the weight $c_{i,j,\varphi_i,\varphi_j,a}$ of this edge in G^c, can be defined:

$$c_{i,j,\varphi_i,\varphi_j,a} = \mu_a(s) = \mu_a(P) \tag{2.15}$$

The extension for a discrete path P with higher number of nodes follows the same build-up principle of primitives: for a path $P = \{(i_1, \varphi_{i_1}), \ldots, (i_n, \varphi_{i_n})\}$, the cost is defined as,

$$\mu_a(P) = \sum_{j=1}^{n-1} \mu_a(\{(i_j, \varphi_{i_j}), (i_{j+1}, \varphi_{i_{j+1}})\}), \tag{2.16}$$

that is, the sum of the individual costs of each neighboring segment present in P. For example, consider a path $P = \{(1,1), (2,3), (3,5), (4,1)\}$, then its cost would be the sum of all middle segments:

$$\mu_a(P) = c_{1,4,1,1,a} = c_{1,2,1,3,a} + c_{2,3,3,5,a} + c_{3,4,5,1,a}.$$

2.4 PATH CONSTRUCTION

The ShorPaP can be first enunciated in its continuous form, agreeing with the original continuous PLTAM problem. For an initial point p_i and a final point p_f, the problem consists in finding the least costly path $s \in \mathcal{S}(p_i, p_f)$, while having the required minimum speed v_a^{min},

$$\min_{s \in \mathcal{S}(p_i, p_f)} \mu_a(s), \tag{2.17}$$

or, in a more explicit way and closer to what is usually found in optimal control literature (KIRK, 2004),

$$\begin{aligned} \min_{s\in\mathcal{S}(p_i,p_f)} & \int_0^T L_a(u_a(t))dt \\ s.t. & \\ & \dot{q}_a(t) = f(t, q_a(t), u_a(t)), \\ & 0 = g_a(t, q_a(t)), \\ & h_a(t, q_a(t)) = s(t), \\ & |\dot{s}(t)| > v_a^{min}, \end{aligned} \tag{2.18}$$

where all variables preserve the meaning they had previously on (2.12), and for a path that needs to be built knowing the start and end orientations, the constraints function g_a would feature this requirement.

Translating this idea into a segmentation-aware form, let P be a variable sized node path with known start and end nodes, (i, φ_i) and (f, φ_f), $P = \{(i, \varphi_i), \ldots, (f, \varphi_f)\}$. The ShorPaP can then be solved by finding in G^c which nodes minimize the cost between the start and end nodes,

$$\min_{P\subset N^{c+}} \mu_a(P), \tag{2.19}$$

where all constraints and timing concerns are assumed to be solved by the correct use of lattice gap, primitives choice and positioning. Thus, the computational complexity of computing the solution of (2.18) or (2.17), which usually involve solving a boundary-valued differential partial equation (KIRK, 2004), is traded with a graph minimal path problem, which usually involve techniques with known time and computational resources consumption (CORMEN, 2009).

Two major ways of solving (2.14) is by treating through graph search techniques or modeling it in a combinatorial manner via a Mixed Integer Linear Programming (MILP) model. One big setback of the latter is that it is computationally difficult: as the size of nodes in G^c grows, there are exponentially more possible solutions to seek for a minimum. Problems with this characteristic are usually referred as having NP difficulty (CORMEN, 2009), meaning that bigger instances of this problem cannot be solved within reasonable time and computational resources, as of 2018, since the debate of whether P equals NP still has no solution (WOEGINGER, 2018). Fortunately, the former approach do not suffer from this difficulty, and is able to solve this problem optimally, or at least find a very good solution, by looking at the graph directly and exploiting its connective nature, instead of tackling the combinatorial problem. For example, he Bellman-Ford algorithm has $\mathcal{O}\left(\left(\dim E^{c+}\right)^2\right)$, i.e. polynomial worst case complexity (BELLMAN, 1958), compared to MILP exact methods such as branch-and-bound which have hard to estimate complexities.

The best known ShorPaP solution algorithm to date in a weighted graph, such as G^c, where all weights are positive and no easy guide heuristic is known between any two nodes, is the uniform cost search algorithm, both eager and lazy versions, the former being commonly known as Dijkstra's algorithm (FELNER, 2011; DIJKSTRA, 1959). Both are presented here, in Algorithms 1 and 2, respectively, both being orientation-aware for cohesive construction of optimal paths and able to return all shortest paths from a given starting node to any other. Their main differences reside on the trade-off between memory and processing use: even thought the lazy version would be superior in all ways in a theoretical view (FELNER, 2011), manipulating a list structure takes computational effort, so the total computation time suffers accordingly, whereas the eager has all nodes already loaded from start. Any variation of these algorithms which can efficiently reuse previous computed shortest paths in broader way still did not find widespread study in the related literature, and may be a research topic on its own.

With the optimal paths coded in values and parents tree, constructing a continuous path is done by starting from the leaves of the parents tree and backtracking until the starting state is reached. A final concern is dealing with b^R, due to its out-of-grid nature. The approach used here is to link the "closest" node in N^c to b^R, thus discarding the need to accommodate an extra orientation exclusively related to b^R, then considering that the agents start from this immediate node, rather than in b^R itself, subtracting this small trip between the immediate node and b^R from the agent limit β_a. For example, suppose that after segmentation is done, the closest node to b^R, for agent class a, is i and that the orientation that causes the least cost from i to b^R is φ_i. Then, all agents of class a are considered to start at this states (i, φ_i), while their capacity is reduced by the cost of both going to and from b^R, say, termed c^R, $\beta_a := \beta_a - 2c^R$.

The results of the path construction stage are the set of all interest points, N^d, and the *decision graph*, $G^d = (N^{d+}, E^{d+})$, which is densely connected by the resulting shortest paths computed. Due to this dense connection, it follows, for easier posterior treatment, that $N^{d+} = N^d \times \Phi$.

Algorithm 1 Eager Uniform Cost Search (Dijkstra's algorithm)

Require: $G^c = (N^{c+}, E^{c+})$, a, $c_{i,j,\varphi_i,\varphi_j,a}$, (i_s, φ_s)

$Q \leftarrow N \times \Phi \setminus \{(i_s, \varphi) | \varphi \in \Phi\}$

$Values \leftarrow \{(i, \varphi) \rightarrow \infty | \forall i \in N; \varphi \in \Phi\}$

$Values[(i_s, \varphi_s)] \leftarrow 0$

$Parents \leftarrow \{(i, \varphi) \rightarrow -1 | \forall i \in N; \varphi \in \Phi\}$

while $Q \neq \emptyset$ **do**

  $(i, \varphi_i) \leftarrow$ MinimumValue$(Q, Values)$

  for $(j, \varphi_j) \in$ Neighbors$((i, \varphi_i))$ **do**

    if $Values[(j, \varphi_j)] > Value[(i, \varphi_i)] + c_{i,j,\varphi_i,\varphi_j,a}$ **then**

      $Values[(j, \varphi_j)] \leftarrow Value[(i, \varphi_i)] + c_{i,j,\varphi_i,\varphi_j,a}$

      $Parents[(j, \varphi_j)] \leftarrow (i, \varphi_i)$

    end if

  end for

  $Q \leftarrow Q \setminus \{(i, \varphi_i)\}$

end while

return $Values$, $Parents$

Algorithm 2 Lazy Uniform Cost Search

Require: $G^c = (N^{c+}, E^{c+})$, a, $c_{i,j,\varphi_i,\varphi_j,a}$, (i_s, φ_s)

$Q \leftarrow \{(i_s, \varphi_s)\}$

$C \leftarrow \emptyset$

$Values \leftarrow \{(i_s, \varphi_s)] \rightarrow 0\}$

$Parents \leftarrow \{(i, \varphi) \rightarrow \emptyset\}$

while $Q \neq \emptyset$ **do**

  $(i, \varphi_i) \leftarrow$ ExtractMinimum$(Q, Values)$

  $C \leftarrow C \cup \{(i, \varphi_i)\}$

  for $(j, \varphi_j) \in$ Neighbors$((i, \varphi_i)) \setminus C$ **do**

    $v \leftarrow Values[(i, \varphi_i)] + c_{i,j,\varphi_i,\varphi_j,a}$

    if $(j, \varphi_j) \notin Q$ **then**

      $Q \leftarrow Q \cup \{(j, \varphi_j)\}$

      $Values[(j, \varphi_j)] \leftarrow v$

      $Parents[(j, \varphi_j)] \leftarrow (i, \varphi_i)$

    else if $(j, \varphi_j) \in Q$ and $v < Values[(j, \varphi_j)]$ **then**

      $Values[(j, \varphi_j)] \leftarrow v$

      $Parents[(j, \varphi_j)] \leftarrow (i, \varphi_i)$

    end if

  end for

end while

return $Values$, $Parents$

2.5 OPTIMAL ASSIGNMENT

Consider the set containing all possible cyclic node paths in G^d starting and ending b^R, $\mathcal{P}(b^R)$, the discrete equivalent of $\mathcal{S}(b^R)$, The objective is to assign enough cycles $P \in \mathcal{P}(b^R)$ for different agent classes, $a \in I^a$, satisfying both demand and capacity constraints. One way to systematically treat this discrete combinatorial optimization is through an MILP model.

2.5.1 Mixed Integer Linear Model

Let $x_{i,j,\varphi_i,\varphi_j,a}$ be an integer decision variable that decides how many agents of class $a \in I^a$ move from $(i,\varphi_i) \in N^{d+}$ to another state $(j,\varphi_j) \in N^{d+}$, this integer definition emanating from the fact that agents of the same class may have intersecting paths to reach their respective goals. The objective is minimization of all paths costs,

$$\min_{x} \sum_{\substack{i\in N^d, j\in N^d \\ \varphi_i\in\Phi,\varphi_j\in\Phi \\ a\in I^a}} c_{i,j,\varphi_i,\varphi_j,a} x_{i,j,\varphi_i,\varphi_j,a}, \tag{2.20}$$

with four constraints. First, only connected and cohesive paths, whatever the chosen primitives (Figure 2.9), for every agent class a, are allowed,

$$\sum_{\substack{i\in N^d \\ \varphi_i\in\Phi}} x_{i,j,\varphi_i,\varphi_j,a} - \sum_{\substack{k\in N^d \\ \varphi_k\in\Phi}} x_{j,k,\varphi_j,\varphi_k,a} = 0, \quad \forall j \in N^d; \varphi_j \in \Phi; a \in I^a. \tag{2.21}$$

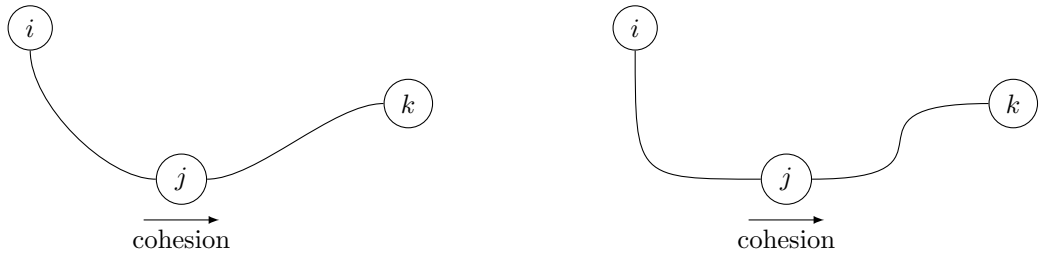

Figure 2.9: Two different continuous reconstruction with cohesive node links.

Second, all demands must be covered by the agents able to supply that demand visiting the cell,

$$\sum_{\substack{i\in N^r, a\in I^a \\ \varphi_i\in\Phi,\varphi_j\in\Phi}} v_{j,a,o} x_{i,j,\varphi_i,\varphi_j,a} \geq \alpha^d_{j,o}, \quad \forall j \in N^r; o \in I^o, \tag{2.22}$$

where N^r, $\alpha^d_{j,o}$ are constructed from I^r, $\alpha_{j,o}$, and $v_{j,a,o}$ is constructed from $\mu^d_{a,o}$ in accordance to the segmentation scheme, preserving (2.2) demand properties. On the other hand, considering that the decision graph retains only nodes to be visited, and that the primary function of this demand constraint is to ensure that they are all visited and covered, a further reduction can be made,

$$\sum_{\substack{i\in N^d, a\in I^a \\ \varphi_i\in\Phi,\varphi_j\in\Phi}} \delta_{a,o} x_{i,j,\varphi_i,\varphi_j,a} \geq \delta_{j,o}, \quad \forall j \in N^d \setminus \{b^R\}; o \in I^o, \tag{2.23}$$

where $\delta_{a.o}$ is 1 if an agent class a can do some operation and 0 otherwise, the same applies for $\delta_{j,o}$ in relation to a node j. This observation removes the necessity to compute and store $v_{j,a,o}$, $\alpha^d_{j,o}$ for all interest nodes and operations.

For the remaining subtour and capacity constraints, let $y_{i,j,\varphi_i,\varphi_j,a}$ be a variable that represents if any class a agent transverses from (i,φ_i) to (j,φ_j), then, it can be defined in relation to x,

$$y_{i,j,\varphi_i,\varphi_j,a} = \min(1, x_{i,j,\varphi_i,\varphi_j,a}) \tag{2.24}$$

and let $\mathcal{P}^C(b^R)$ be the set of all cyclic paths that do not contain the base. Then, every agent class that is in a cycle without the base must necessarily exit this path to somewhere else, meaning that $P \in \mathcal{P}^C(b^R)$ must not sum up to the total of nodes that it contains,

$$\sum_{\substack{(i,\varphi_i)\in P \\ (j,\varphi_j)\in P}} y_{i,j,\varphi_i,\varphi_j,a} \leq \dim P - 1, \quad \forall P \in \mathcal{P}^C(b^R); a \in I^a, \tag{2.25}$$

and the fourth, capacity constraint,

$$\sum_{\substack{(i,\varphi_i)\in P \\ o\in I^o}} \delta_{a,o} c^n_{i,o} + \sum_{(i,\varphi_i)\in P} c_{i,i^+,\varphi_i,\varphi_{i^+},a} y_{i,i^+,\varphi_i,\varphi_{i^+},a} \leq \beta_a, \quad \forall P \in \mathcal{P}(b^R); a \in I^a, \tag{2.26}$$

where $c^n_{i,o}$ is the cost of performing operation o at node i, assuming that the cost of doing the operation is indifferent of the agent doing it, β_a are the same limits used in the continuous case, and the sum signifies a cyclical sequential addition, with i^+ being the next node of i in P, i.e. first and second, second and third, and so on.

Since constraints (2.25) and (2.26) are equivalent to subtour elimination constraints commonly found in Traveling Salesman Problems (TSPs) and VRPs (GOLDEN; RAGHAVAN; WASIL, 2008; TOTH; VIGO, 2002), it would be possible to replace these for equivalent MTZ (DESROCHERS; LAPORTE, 1991) or commodity-flow (LANGEVIN; SOUMIS; DESROSIERS, 1990) constraints. Yet, doing such modifications diminishes some computational properties of the MILP model, namely, it makes the relaxation of the problem itself weaker (TOTH; VIGO, 2002), fixing the best solution obtainable at a margin of the best solution obtainable with the model shown here. Moreover, recent research points out that this is true for any MTZ-like constraint, notwithstanding its form (BEKTAŞ; GOUVEIA, 2014).

Another related restatement, in attempt to ease the exponential number of inequalities, is to include a temporal index τ directly. That is, make x time-aware, $x_{i,j,\varphi_i,\varphi_j,a,\tau}$, which would trade a higher number of variables for removal of subtour restrictions, since a solution which makes a connected route go unnecessarily backwards certainly is not optimal. Further examination shows that this trade-off cannot in fact be achieved, because every agent is required to start and end its route at the base b^R within any time interval; and that requires the model to have a subtour-elimination alike (2.25) for any time interval, actually increasing the number of constraints.

Also, the MILP model can be "relaxed" (not to be confused with relaxed solutions) for lighter computational costs when using exact methods, e.g. branch-and-bound, by simply removing the binary that depends variable y and using x in its place. This turns large-scale complex situations intractable, but can improve the running time and memory requirements for smaller scenarios. To see how this modeling difference affects the problem tractability, consider that at certain moment, a path P, with 6 states, for some agent class a, connected to b^R, is analyzed, and that a subpath P', with 3 states, inside P without connection to b^R (Figure 2.10) has been analyzed beforehand. Since the x decision variable cannot sum more than 2 in P', if $x > 2$ in P while P is feasible capacity-wise, then this solution that should be feasible with the presence of y, would be accused of infeasibility in the relaxed model. A similar argument can show that a solution that respects the capacity limits of each agent class can be deemed infeasible from the relaxed model perspective. Thus, it is a useful tool only when the optimal routes intersections are known to be sparse.

To solve this MILP model, it is possible to use general-case algorithms available to the standard form for these models, where x and c must be flattened to a single row vector and all inequalities must also be flattened to obey a matrix inequality,

$$\min c^T x \qquad s.t. \quad Ax \leq b, \tag{2.27}$$

equalities can then be introduced by means of slack or surplus variables. If one uses MILP solvers, Equation (2.27) is the object of solution, mainly by branch-and-bound techniques. Another approach, considering this problem NP-Hard complexity, is to construct specialized heuristics that solve the MILP model indirectly through some special structure of the problem for increased time and resources performance, whilst finding acceptable valued solutions.

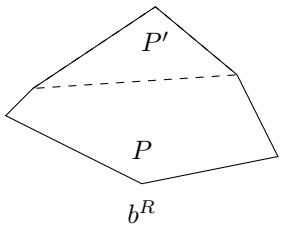

Figure 2.10: Example of possible problems with relaxed model limits.

2.5.2 Admissible Score Clustering heuristics family

For problems that resemble VRPs, a viable solution is to first cluster nodes someway and then optimize the route in this cluster, which consists to solving a local instance of a TSP, effectively separating it into a general assignment problem coupled with many TSPs. The algorithm developed here, Admissible Score Clustering (ASC), takes inspiration from tabu-search meta-heuristics (GLOVER, 1986) and other cluster-and-route heuristics already present in the literature (FISHER; JAIKUMAR, 1981). Its conception is modular and treats routes in the graph directly through transformations in order to achieve better performance, making every solution obtained also feasible. Unsurprisingly, graph matching problems, where graph transformations are normally defined and studied, are also NP-hard (ZAGER; VERGHESE, 2008). The pseudo-code for general ASC is displayed at Algorithm 3 and its description follows.

Let a tour w be the junction of a path $P \in \mathcal{P}(b^r)$ with an agent class $a \in I^a$, $w = (P, a)$. The starting solution W is one very far from optimal, composed by single node tours for each of the agent classes. If this initial solution cannot be made feasible by removal of some tours, the overall problem is infeasible. From this initial solution the main loop starts: all tours are locally optimized as a TSP followed by determination of the highest variation feasible transformation T from the transformations set $\mathcal{T}$, (discussion of the local optimization and transforms $\mathcal{T}$ are deferred because of their interchangeability), once this best transform, T^*, is determined and applied, its score is updated according to the solution variation. If a local minimum is reached, it is stored if better than the best solution or discarded otherwise, then, a new solution with a random feasible subset of all tours visited so far is built for another iteration, but not before applying a forgetting factor to the scores computed up to that moment. The algorithms stops if no better solution is attained after $maxRepeats$ local minima were found or $maxIters$ iterations have elapsed. Here, a simple scoring system which consists of adding one to every tour at every iteration was used, and the forgetting vector is set to be make all scores lose 1% of their value at each iteration.

Four transforms were defined in this work in a way that they can return favorable variations for every solution possible; two of these transform are macroscopic transformations while the other two are microscopic. Explicitly,

1. Tour Merge: merge in a tail-head fashion two tours from the solution. Illustrated in Figure 2.11.
2. Tour Removal: completely erase a tour from the solution. Illustrated in Figure 2.12.
3. Node Transfer: transfer the nodes between two routes that makes the most cost reduction along their reorientation. Illustrated in Figure 2.13.
4. Node Removal: completely remove a node from a tour in a solution. Illustrated in Figure 2.14.

It must be remarked that the new tours produced by each transform retain the previous agent class, limiting manipulation of classes in use by a solution to tour and node removals. Also, transfer is chosen in place of insertion as it may result in a favorable transform, whereas insertion would always be unfavorable, requiring special treatment. The following variation deductions, whenever applicable, use the same variables of the illustrations as a visual aid.

Transference variation applied to two tours w_1, w_2 in a solution is given by the best sum of node insertion and removal in the direction $w_1 \to w_2$ or in $w_2 \to w_1$,

$$\Delta^{transfer}_{w_1 \to w_2} = \min_{k \in N(w_1)} \left(\Delta^{insertion}_{w_2,k} + \Delta^{removal}_{w_1,k} \right),$$

Algorithm 3 ASC

Require: N^d, Φ, I^a, c, $Constraints$, $\mathcal{T}$, $maxRepeat$

$W \leftarrow \textsc{InitialSolution}(N^d, \Phi, I^a, c, Constraints)$
$Scores \leftarrow \emptyset$
$repeat \leftarrow 0$
$W^{best} \leftarrow W$
$Tours \leftarrow \{representation(w) \rightarrow w | w \in W\}$
while $repeat < maxRepeat$ **do**
  for $w \in W$ **do**
    $w \leftarrow \textsc{TourOptimization}(w, c)$
    $Tours[representation(w)] \leftarrow w$
  end for
  $score \leftarrow 0$
  $W^{next} \leftarrow W$
  $T^* \leftarrow Identity$
  for $T \in \mathcal{T}$ **do**
    $\Delta^T, W^T \leftarrow T(W, \Phi, c)$
    $score^T \leftarrow \textsc{ApplyScore}(\Delta^T, T, W, W^T, Scores)$
    if $score < score^T$ and $Constraints(W^T)$ **then**
      $W^{next} \leftarrow W^T$
      $T^* \leftarrow T$
      $score \leftarrow score^T$
    end if
  end for
  if $score > 0$ **then**
    $Scores \leftarrow \textsc{UpdateScores}(T^*, W, W^{next}, Scores)$
    $W \leftarrow W^{next}$
  else
    if $\textsc{SolutionValue}(W) < \textsc{SolutionValue}(W^{best})$ **then**
      $repeat \leftarrow 0$
      $W^{best} \leftarrow W$
    end if
    $repeat \leftarrow repeat + 1$
    $W \leftarrow \emptyset$
    while not $Constraints(W)$ **do**
      $W \leftarrow W \cup \textsc{Random}(Tours)$
    end while
  end if
  $Scores \leftarrow \textsc{ForgetScores}(Scores)$
end while
return W^{best}

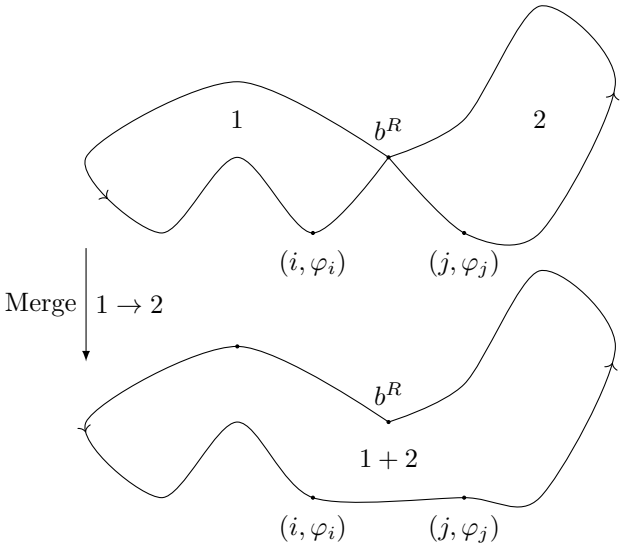

Figure 2.11: Illustration of the tour merge transformation.

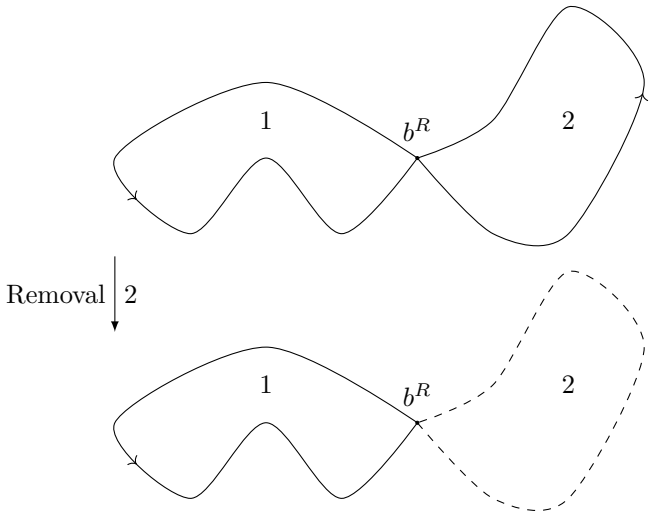

Figure 2.12: Illustration of the tour removal transformation.

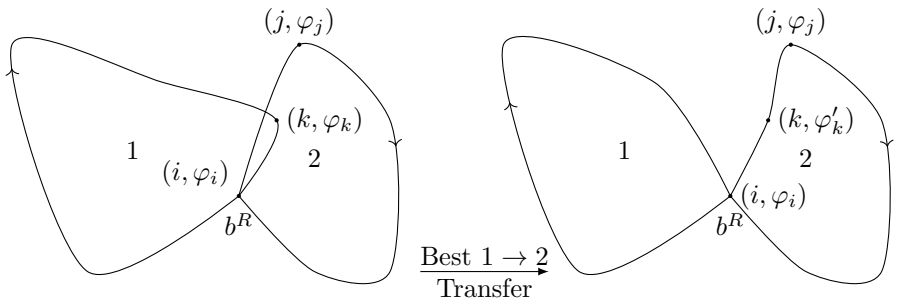

Figure 2.13: Illustration of the node transfer transformation.

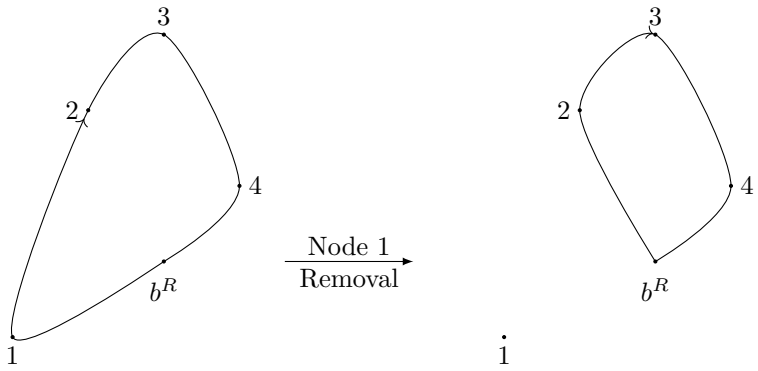

Figure 2.14: Illustration of the node removal transformation.

where N is a function that returns the ordered node set from a tour, i.e. $N(w) \subset N^d$, and the minimization aligns with the search for the most favorable (most negative) transform. Now, the insertion variation for adding a node k in w, is given by subtracting the edge that existed before insertion and summing most favorable state addition in the route w with agent class a,

$$\Delta_{w,k}^{insertion} = \min_{\varphi_k \in \Phi} (c_{i,k,\varphi_i,\varphi_k,a} + c_{k,j,\varphi_k,\varphi_j,a}) - c_{i,j,\varphi_i,\varphi_j,a},$$

while removal variation of a node k is simply computed by subtraction of the edges that were in the tour before-hand and sum of the new formed edge in the route w with agent class a,

$$\Delta_{w,k}^{removal} = c_{i,j,\varphi_i,\varphi_j,a} - (c_{i,k,\varphi_i,\varphi_k,a} + c_{k,j,\varphi_k,\varphi_j,a}).$$

For a tour w with path P and agent class a to be removed from any solution W, the variation is naturally subtraction of its cost,

$$\Delta_{w}^{removal} = -\mu_a(P),$$

whereas merge cost variation is given by the minimum of two possible head-tail combinations for two merging tours $w_1, w_2 \in W$. Let (i, φ_i) be the final state of w_1 before the base state (b^r, φ_b), and let (j, φ_j) be the state of w_2 just after base exit (b^R, φ_b); the variation of this cost is given by,

$$\Delta_{w_1 \to w_2}^{merge} = c_{i,j,\varphi_i,\varphi_j,a} - (c_{i,b^R,b,\varphi_b,a} + c_{b^R,j,\varphi_i,\varphi_j,a}),$$

merge direction $w_2 \to w_1$ is defined symmetrically.

Regarding local tour optimization, one possibility is to solve the orientation-aware TSP MILP model directly, where techniques prevalent in the literature could be adapted to accommodate orientations. Here two heuristics where tested, one inspired by ASC itself and one faintly based on Ant Colony heuristics (DORIGO; BLUM, 2005) and construction methods. The former, named Order and Orientation Switching (OOS) is shown in Algorithm 4, with its transforms $\mathcal{T}^w$ consisting of reordering states or reorienting them, while the latter, named Restarting Sequential Construction (ReSC) is shown in Algorithm 5 and consists in choosing two starting states randomly and trying to construct the best tour possible based on this starting tour, both local heuristics make use of scoring for local attractor escaping. The ReSC connection with Ant Colony Heuristics is given by its constructive nature and the scoring system: like pheromones, the scores guide the balance between exploration and exploitation of the local optimization, but also are forgotten as the procedure goes on, just like the pheromones become less pronounced.

Likewise $\mathcal{T}$ required explicit computation in ASC, acquiring the costs for $\mathcal{T}^w$ is a necessity for OOS; but since reorientation is a rather simple transform, only node switching is enunciated next. Switching requires consideration of two separate cases: whether they are follow-ups in their tour or not. If they are not, let (i, φ_i) and (j, φ_j) be states of w and assume without loss of generality that i comes first than j in w; let (i^-, φ_i^-) and (j^-, φ_j^-) be preceding states of i and j, similarly define i^+ and j^+. The variation $\Delta_{i,j}^{switch}$ is given by:

$$\Delta_{j \to i}^{switch} = \min_{\varphi_j' \in \Phi} \left(c_{i^-,j,\varphi_i^-,\varphi_j',a} + c_{j,i^+,\varphi_j',\varphi_i^+,a} \right) - \left(c_{i^-,i,\varphi_i^-,\varphi_i,a} + c_{i,i^+,\varphi_i,\varphi_i^+,a} \right),$$

$$\Delta_{j \to i}^{switch} = \min_{\varphi_i' \in \Phi} \left(c_{j^-,i,\varphi_j^-,\varphi_i',a} + c_{i,j^+,\varphi_i'',\varphi_j^+,a} \right) - \left(c_{j^-,j,\varphi_j^-,\varphi_j,a} + c_{j,j^+,\varphi_j,\varphi_j^+,a} \right),$$

Algorithm 4 OOS

Require: w, Φ, c, $\mathcal{T}^w$, $maxRepeats$

$Scores \leftarrow \emptyset$

$repeat \leftarrow 0$

$w^{best} \leftarrow w$

while $repeat < maxRepeat$ **do**

$score \leftarrow 0$, $w^{next} \leftarrow w$, $T^* \leftarrow Identity$

for $T \in \mathcal{T}^w$ **do**

Δ^T, $w^T \leftarrow T(w, \Phi, c)$

$score^T \leftarrow \textsc{ApplyScore}(\Delta^T, T, w, w^T, Scores)$

if $score < score^T$ **then**

$w^{next} \leftarrow w^T$

$T^* \leftarrow T$

$score \leftarrow score^T$

end if

end for

if $score > 0$ **then**

$Scores \leftarrow \textsc{UpdateScores}(T^*, w, w^{next}, Scores)$

$w \leftarrow w^{next}$

else

if $\textsc{TourValue}(w) < \textsc{TourValue}(w^{best})$ **then**

$repeat \leftarrow 0$

$w^{best} \leftarrow w$

end if

$repeat \leftarrow repeat + 1$

$w \leftarrow \textsc{Shuffle}(w)$

end if

$Scores \leftarrow \textsc{ForgetScores}(Scores)$

end while

return w^{best}

Algorithm 5 ReSC

Require: w, Φ, c, $maxRepeats$

$Scores \leftarrow \emptyset$

$repeat \leftarrow 0$

$w^{best} \leftarrow w$

$N \leftarrow \textsc{ExtractNodes}(w)$

while $repeat < maxRepeat$ **do**

$N^w \leftarrow N$

$w^{next} \leftarrow \{(b^R, \varphi_{b^R}), \textsc{RandomState}(N, \Phi)\}$

while $N^w \neq \emptyset$ **do**

$w^{next} \leftarrow \textsc{BestAddition}(w^{next}, N^w, c, Scores)$

$N^w \leftarrow N^w \setminus w$

$Scores \leftarrow \textsc{UpdateScores}(w^{next}, Scores)$

end while

if $\textsc{TourValue}(w) < \textsc{TourValue}(w^{best})$ **then**

$repeat \leftarrow 0$

$w^{best} \leftarrow w$

end if

$repeat \leftarrow repeat + 1$

$Scores \leftarrow \textsc{ForgetScores}(Scores)$

end while

return w^{best}

$$\Delta_{i,j}^{switch} = \Delta_{j\to i}^{switch} + \Delta_{i\to j}^{switch}.$$

Otherwise, if $i^+ = j$ and $i = j^-$, then due to their path overlap,

$$\Delta_{i,j}^{switch} = \min_{\substack{\varphi'_i\in\Phi \\ \varphi'_j\in\Phi}} \left(c_{i^-,j,\varphi_i^-,\varphi'_j,a} + c_{j,i,\varphi'_j,\varphi'_i,a} + c_{i,j^+,\varphi'_i,\varphi_j^+,a}\right) - \left(c_{i^-,i,\varphi_i^-,\varphi_i,a} + c_{i,j,\varphi_i,\varphi_j,a} + c_{j,j^+,\varphi_j,\varphi_j^+,a}\right).$$

2.6 IMPLEMENTATION ASPECTS

The positioning for segmentation, that is, Equation (2.6), is done via simple linear searches, as they provide a balance between ease of implementation, light computational cost and solution accuracy; this balance regulated by the amount of partitions made in the search interval.

The pertinence function for R, δ_R, can be computed in two ways. If R (or any R_i) is already defined through a function g,

$$R = \{x \in \mathbb{R}^2 \mid g(x) \leq 0\}$$

then it is just a matter of computing the value of g given a desired p to be check for region R. Otherwise, if the shape R is defined by many boundary nodes connected by straight lines, a different approach is necessary to correctly test if p is inside R. A fairly general and efficient algorithm is a specialized form of ray casting: from each candidate point p, a half-line is drawn to either left or right. Let *crosses* be the number of boundaries crosses the half-line made in the region R, then if *crosses* is an odd number, the point p is inside the region, otherwise, it is outside (Figure 2.15). This method also works to check the former function-defined boundary; and while this is theoretically correct, simply using p directly with g greatly improves performance over any other algorithmic method. Asymptotically speaking, the ray casting algorithm is, at least, of linear complexity on the size of boundary nodes; while direct evaluation is always constant in complexity, i.e $\mathcal{O}(n)$ vs $\mathcal{O}(1)$, where $\mathcal{O}(\{)$ is the class of functions that are majored by f (CORMEN, 2009; APOSTOL, 2007). An efficient implementation is available in Algorithm 6.

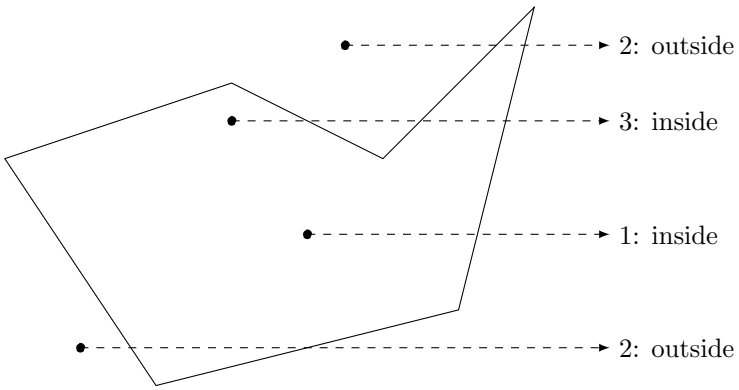

Figure 2.15: Ray casting algorithm visualization.

Those versed in convex optimization may question the existence of some better search tool in case R is a convex shape. Due to the discrete nature of the optimization problem, with many instantaneous oscillations in value, even with a convex region R any guided search method that does not go through, in a resolution-wise sense, the whole displacements domain, is likely to fail finding the cloud point optimization global solution. Figure 2.17 shows a very simple convex region R, covered with circles of radius $\rho^c = 0.5$, along resulting values of δ_R for some variations in the displacement vector d; and the entanglement observed shows the existence of valleys even for simple shapes.

Also, since only the nodes inside R are necessary, a check-up can be made at the segmentation section which retains only the points that are inside R for path construction and consequently assignment. The pertinence function δ_R can be used as a filter once (2.6)-(2.7) is solved, resulting in a lower sized graph with coverage as exemplified by Figure 2.17. Possible spots not covered can be made covered by a lattice refinement in the specific region that contains the spot.

Algorithm 6 Polygon pertinence ray casting algorithm

Require: point p, Boundary B.

```
inside ← False
for p^B ∈ B do
    p' ← NextBoundaryPoint(B, p^B)        ▷ Get succeeding point in the boundary list
    if p^B_y < p_y < p'_y then            ▷ check if y-coord. of p is between those of p^B and p'
        x^ref = ((p'_x − p^B_x)/(p'_y − p^B_y))(p_y − p^B_y) + p^B_x   ▷ Subscripts are axes access.
        if |x^ref − p_x| < tolerance then
            return True                   ▷ if the x coord. is right on the ref. Then its part of the polygon.
        else if x_ref < p_x then
            inside ← ¬inside              ▷ else keep looping and counting.
        end if
    end if
end for
return inside
```

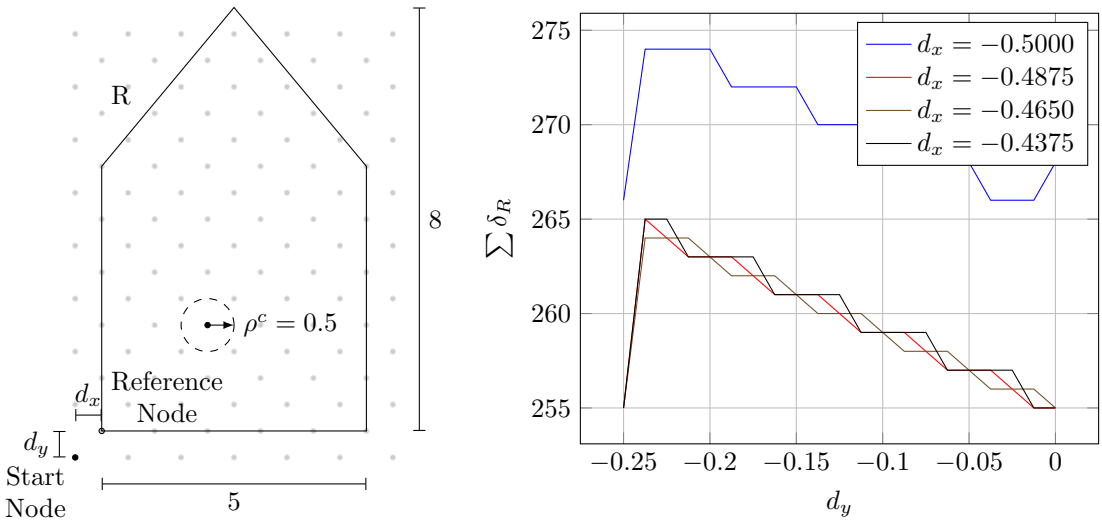

Figure 2.16: Simple convex region and its segmentation objective function.

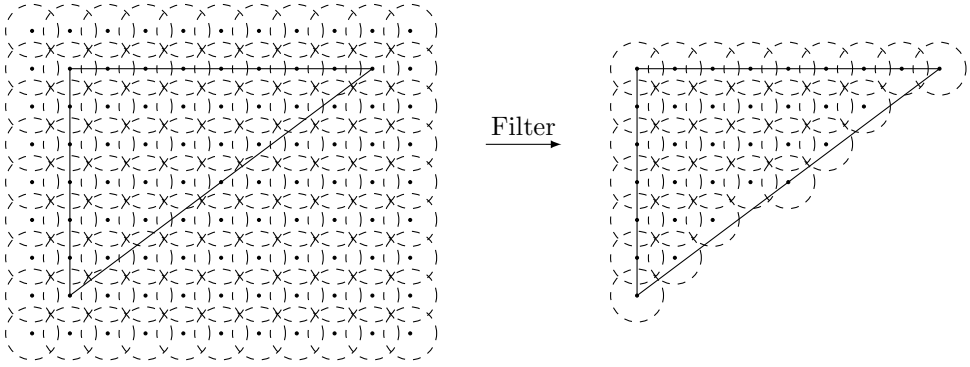

Figure 2.17: Result of filtering for interest nodes after segmentation.

Lastly, a minor possible optimization for the ASC is to simply check specific constraints after each specific transform is executed. For instance, tour removal may violate demand constraint but never the capacity constraint, whereas merge and transfer may violate the capacity constraint, but never the demand constraint.

3 VALIDATION AND RESULTS

3.1 CASE STUDIES

Four missions have been created to test out the merits of this methodology, since there is still a shortage of benchmarks and standardized tests for problems reasonably similar to PLTAMs. All of them have the same underlying region and working agent classes, their main difference being capacity and demand constraints. They are shown in Table 3.1.

Table 3.1: Case studies considered.

Case	Interests	Obstacles	Description	Capacity
1	1 Small	None	Small region to be covered by first operation.	UGV: 100 UAV: 100
2	2 Small	1 Big central	One small region is almost unreachable to UGVs.	UGV: 100 UAV: 100
3	2 Small 1 Big	1 Big central	One small region is almost unreachable to UGVs.	UGV: 100 UAV: 100
4	2 Medium	1 Big central	Both agents have very tight capacity limits.	UGV: 22 UAV: 40

The first and simplest case study is designed to show all the steps of the planner visually. The second is designed to test out the inherent obstacle avoidance of the known map and exercise cooperation between the agents. The third is designed to test out local optimization influence on the solution along agent cooperation. The fourth and last one is designed to test out cooperation and task distribution between the agents, since they have tight capacity limits.

The agent classes are represented by corpuscular entities, one that is a UGV,

$$M^{ground}\dot{v}(t) = -b^{ground}v(t) + u(t),$$

where v is the velocity vector, M^{ground} and b^{ground} are parameters representing lumped inertias and losses, respectively. Velocity damping is a natural lump simplification of terrain effects upon the agent's movement, making it lose momentum, otherwise, without any energy input, the agent would move forward perpetually. The other agent class is a UAV,

$$M^{air}\dot{v}(t) = -b^{air}v(t) + b^{wind}w(t) + u(t),$$

where w is the wind velocity vector, presumably almost constant during operation, and b^{wind} is the lumped wind effect on the UAV, all other variables having equal significance to the UGV. This addition represents in a simple way the wind compliance of a general UAV. The model numerical parameters chosen are displayed in Table 3.2 along the operations each agent class is able to do. In essence, these parameters were chosen to ensure that UGVs have cheaper operation costs than UAVs, while only UAVs can access every area of the map.

Explicitly, the missions attempt to study two major aspects of the planning methodology in addition to demonstrating the workings of the planner: cooperation between the different agent classes to achieve the demand goal, and distribution between agents of the same class to circumvent capacity limitations.

Table 3.2: Agent classes definitions

Agent	Parameters	Operations
UGV	$M^{ground} = 1.0$	1, 2
	$b^{ground} = 0.1$	
UAV	$M^{air} = 2.0$	2
	$b^{air} = 0.05$	
	$b^{wind} = 0.1$	
	$w(t) = [-1, 0]^T$	

These four missions were solved either by exact techniques, as defined by branch-and-bound based ones with lazily instanced constraints, via the commercial GUROBI solver (GUROBI OPTIMIZATION, 2016), and by two heuristics from the ASC family discussed earlier: ASC+OOS and ASC+ReSC. Hardware and software-wise, all overall programs were coded in the open-source language Julia, except by the GUROBI solver which is closed-source, and ran in a common desktop computer. It should also be noted that the relaxed model was used for the exact methods, since preliminary tests without relaxation ran for more than 10 hours without finding their respective solution and that these proposed initial tests have relatively small cover and demand area, making them unlikely to show the intractability discussed in Section 2.5.1. Finally, the stopping criteria for the heuristics used were 300 iterations for the ASC overall, allowing up to 10 local minimum repetitions, and 100 iterations for any local optimization algorithm, allowing again 10 local minimum repetitions before stopping.

3.2 SIMULATION RESULTS

The aggregated results from these solutions can be seen in Table 3.3, with descriptive statistics from 100 heuristic techniques runs and 15 exact technique runs. Additionally, Figures 3.1 through 3.8 show at least one solution for each case covered, remarking that the exact and heuristic solutions for cases 1 and 2 were identical.

Figures 3.1 to 3.4 show all the steps the planner did for case 1 in a demonstrative fashion; starting in the segmentation stage, filtering the interest points, building the shortest paths and finally solving the assignment problem to find the solution. Note that the paths showed for the construction stage (Figure 3.3) are a sample of the total found best paths between the two interest nodes. The second case (Figure 3.5) serves as an example of the obstacle avoidance feature of the planner, by means of path cost invalidation for any such paths that traverse an obstacle for that particular agent class, as seen in this case, where the UGV detours from its optimal path found in case 1 because of the central obstacle, whereas the UAV moves without problems over this same obstacle for a interest region inside it. Again, the resulting solution is identical via both heuristic and exact methods. The other two cases (Figures 3.6 to 3.9) have bigger scales and can be subject to further analysis.

Visually, it can be seen from all Figures representing the solutions that this planner was successful on its objective for creating collaborative continuous paths for the agents. Second, the exact solutions shows that coverage patterns usually set a priori, such as swiping, appear naturally on the reconstructed tours, although the agents exit out of the swipe motion early for increased cost economy while still retaining demand coverage, as evidenced particularly in Figure 3.9.

Some numerical properties of the model and techniques used can be observed from the collected data. For the cooperation mission, where the capacity constraint is greatly relaxed and far from optimal solutions with a single tour for each agent class are easily feasible, exact methods can find the final solution quickly, with about 33% gain in mean time against the two heuristics used, while having 13% and 20% gain in mean optimum value (including tasks costs). On the other hand, the distribution mission, where the capacity was tight and far from optimal tours would be infeasible, exact methods ran for much longer than the heuristics, the mean running time of the former being at least 12500% of the latter, whereas

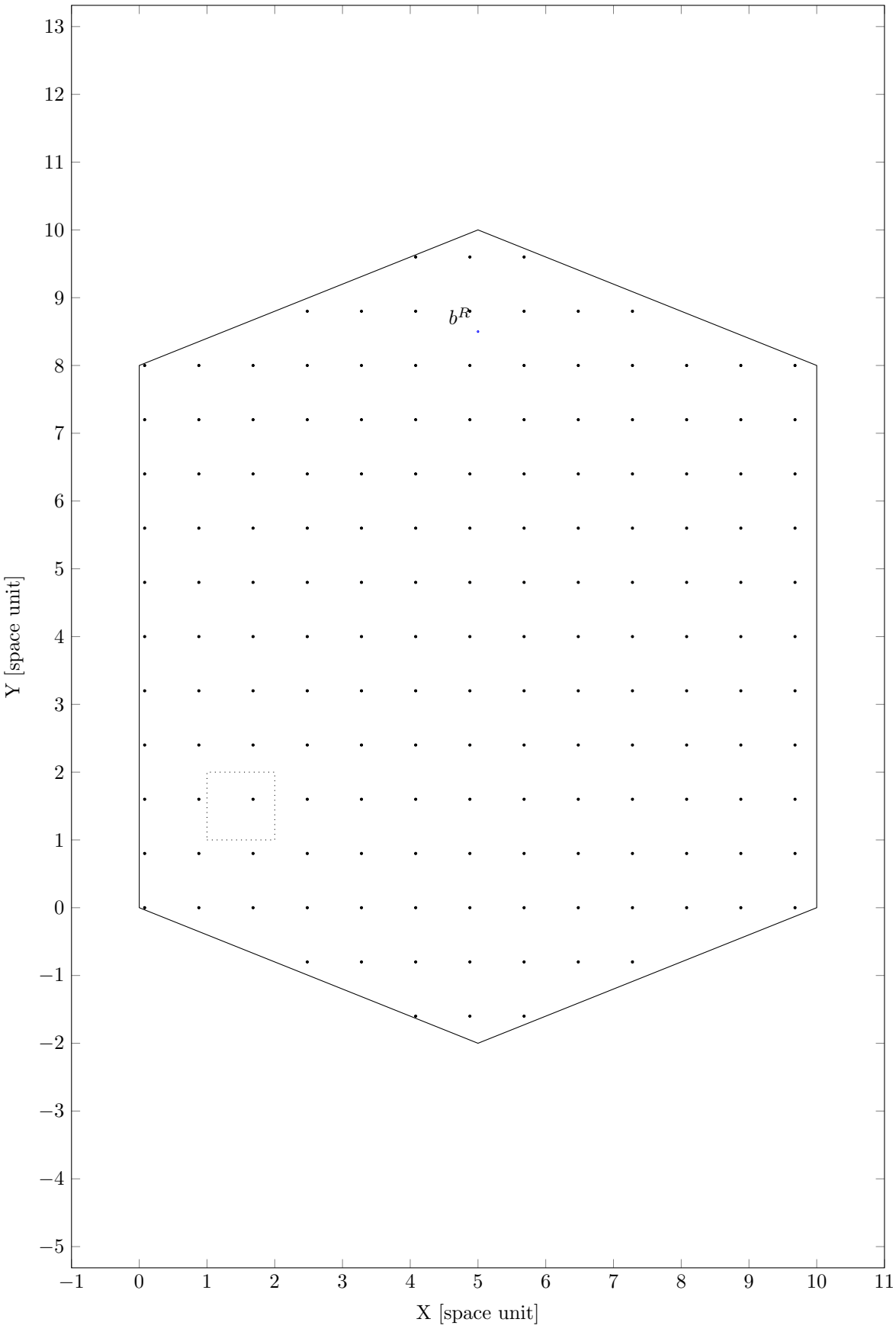

Figure 3.1: Point cloud and *cell graph* of case 1.

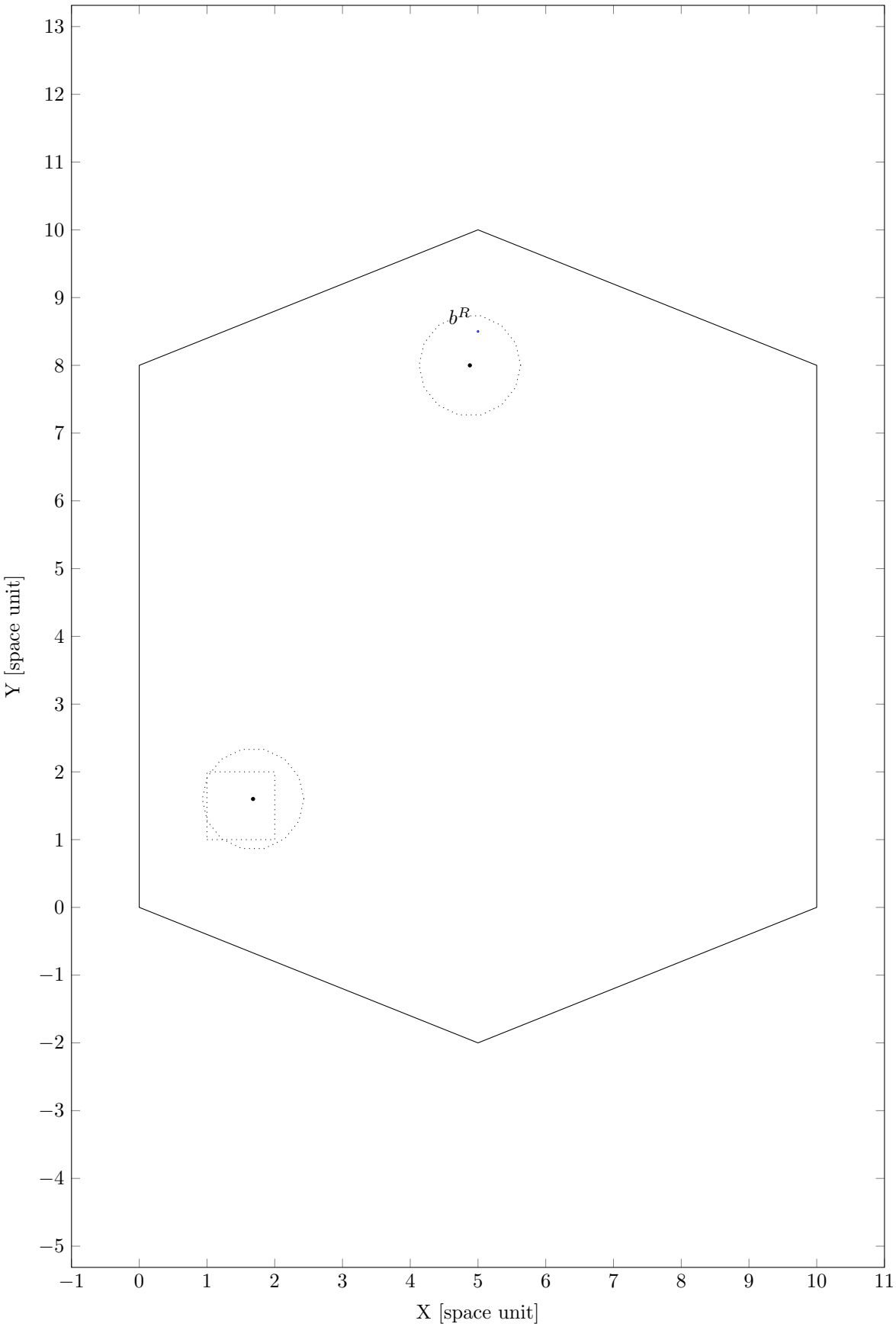

Figure 3.2: Filtered interest nodes for case 1 with their covering radii.

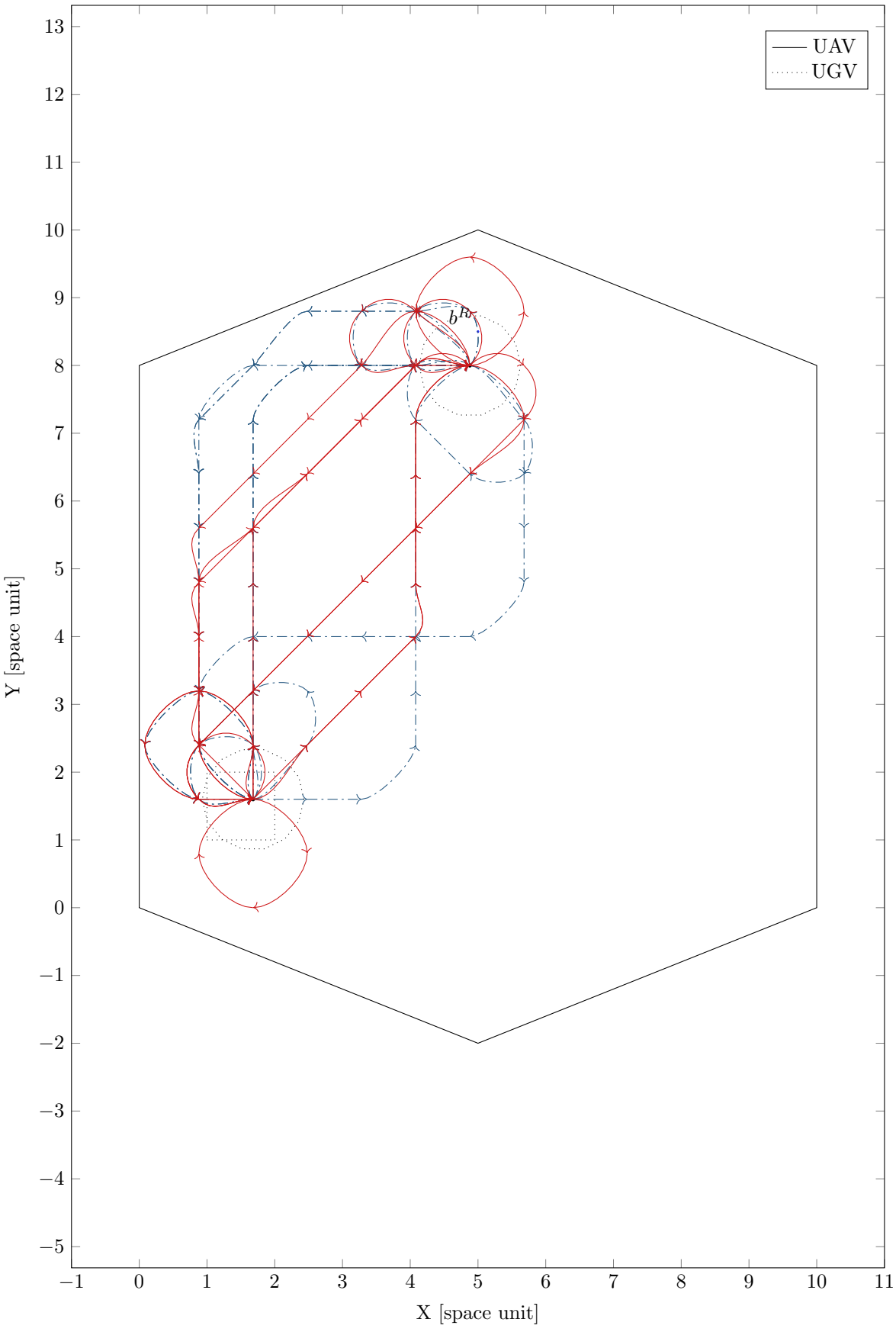

Figure 3.3: Sample of all constructed paths for case 1.

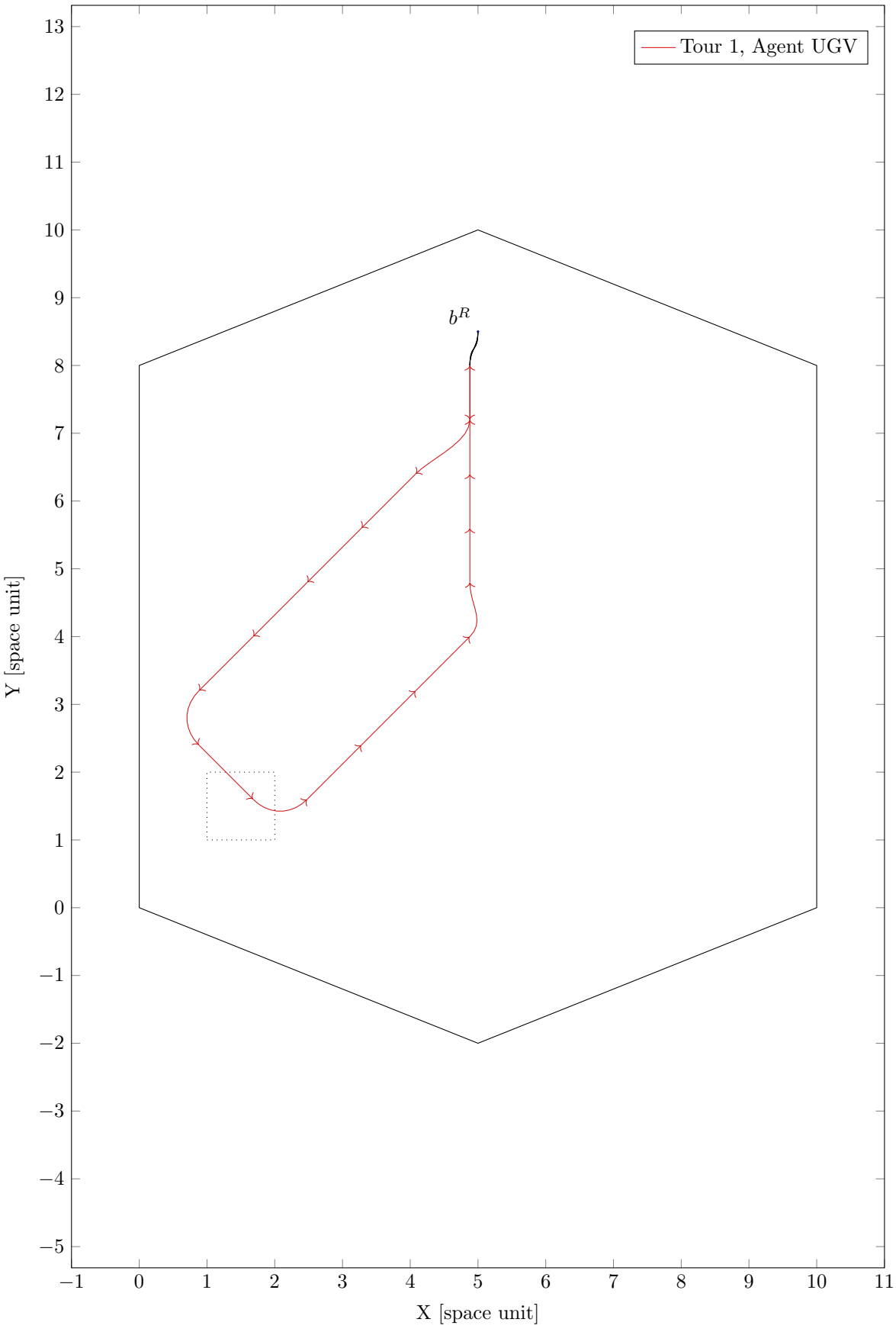

Figure 3.4: Resulting tours for exact and ASC solutions of case 1.

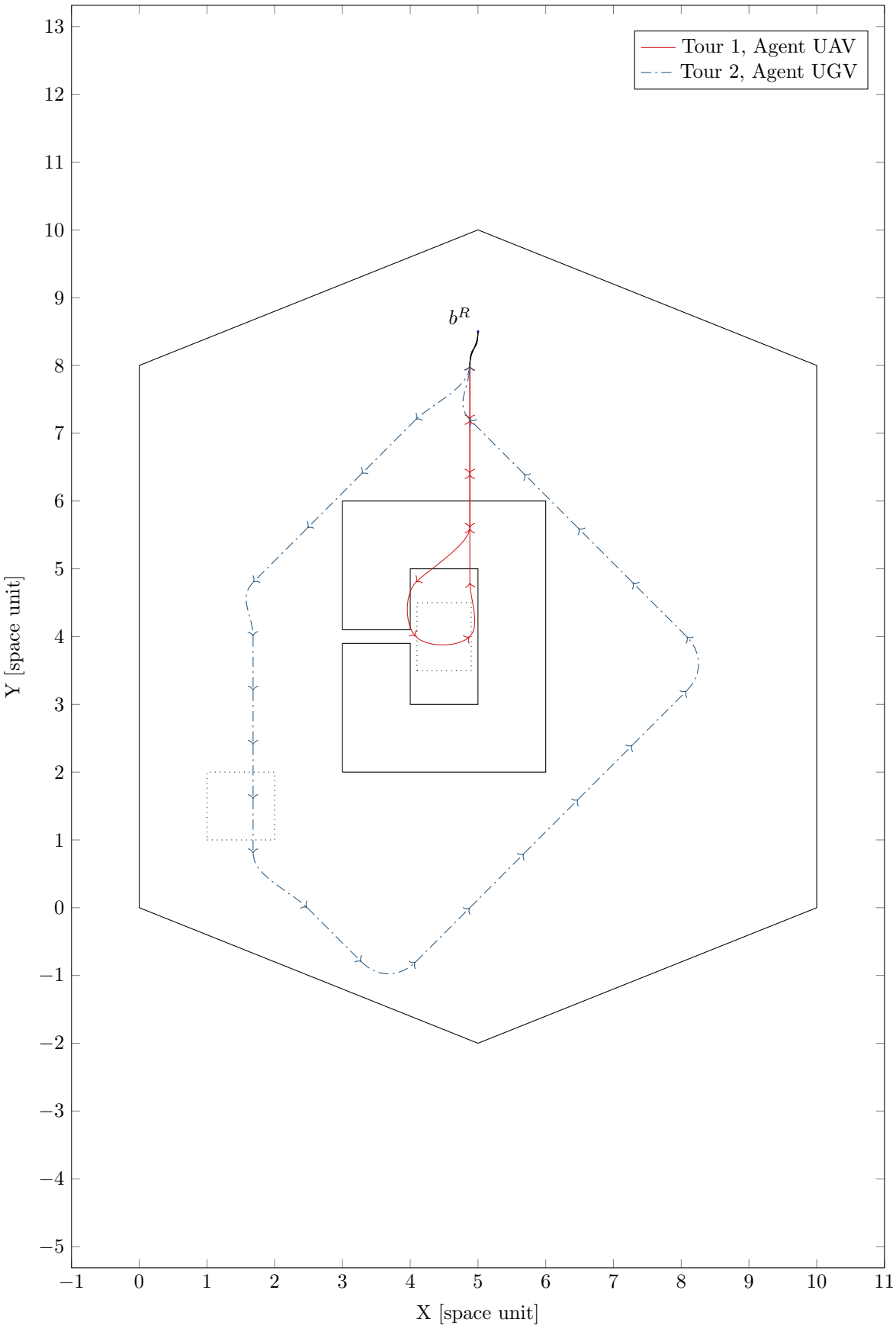

Figure 3.5: Resulting tours for exact and ASC solutions of case 2.

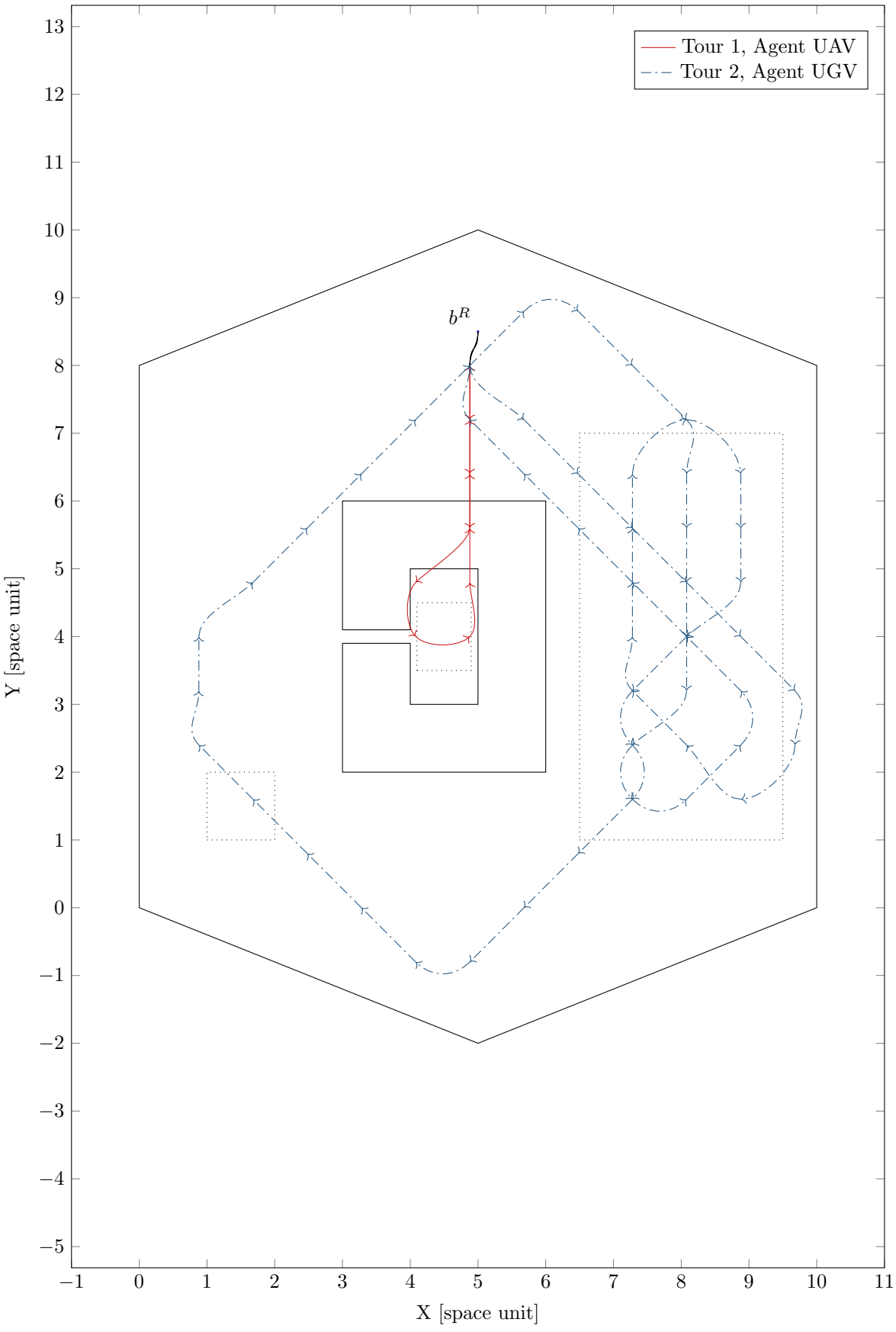

Figure 3.6: Resulting tours for a ASC+OOS solution of case 3.

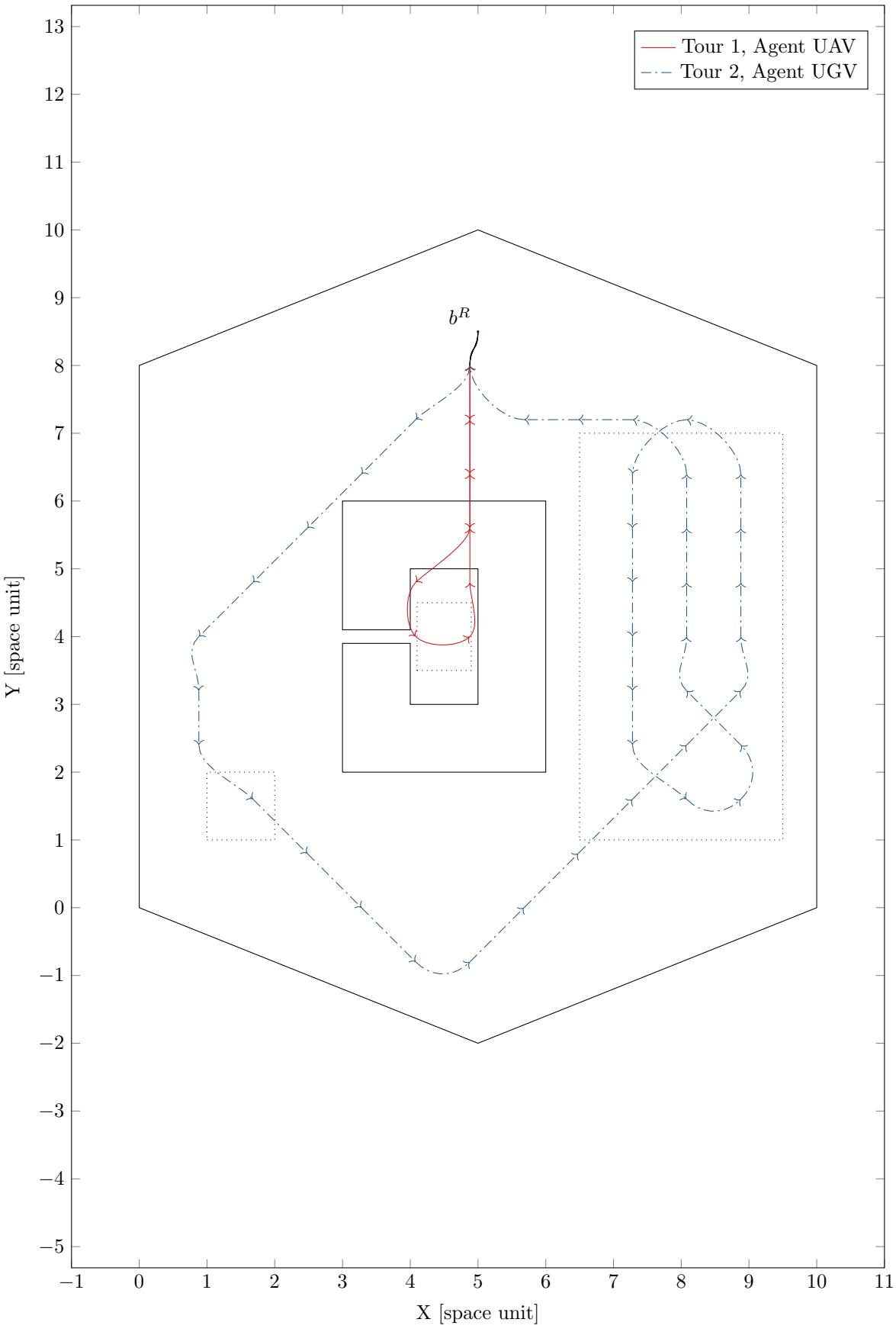

Figure 3.7: Resulting tours for exact solution of case 3.

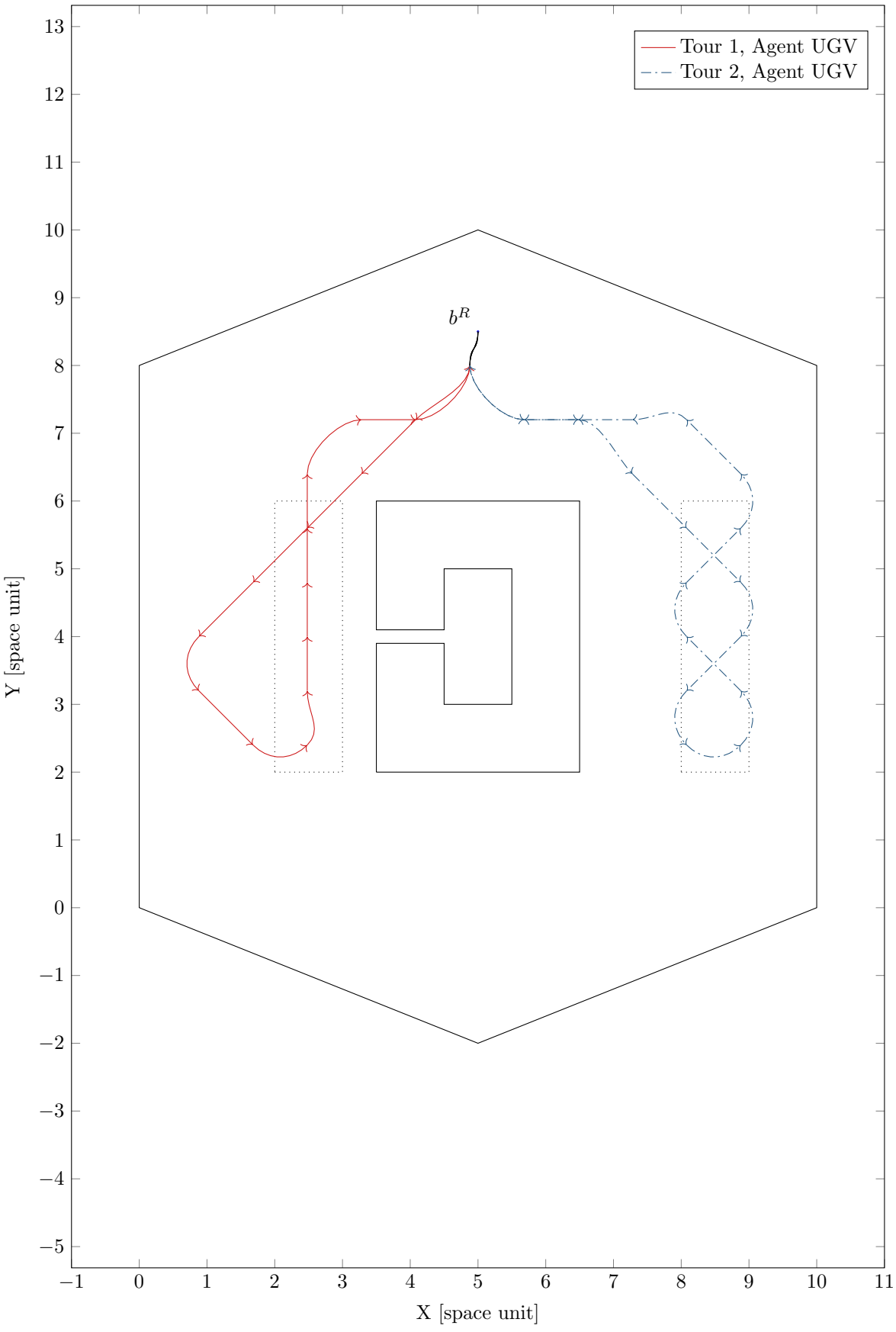

Figure 3.8: Resulting tours for a ASC+OOS solution of case 4.

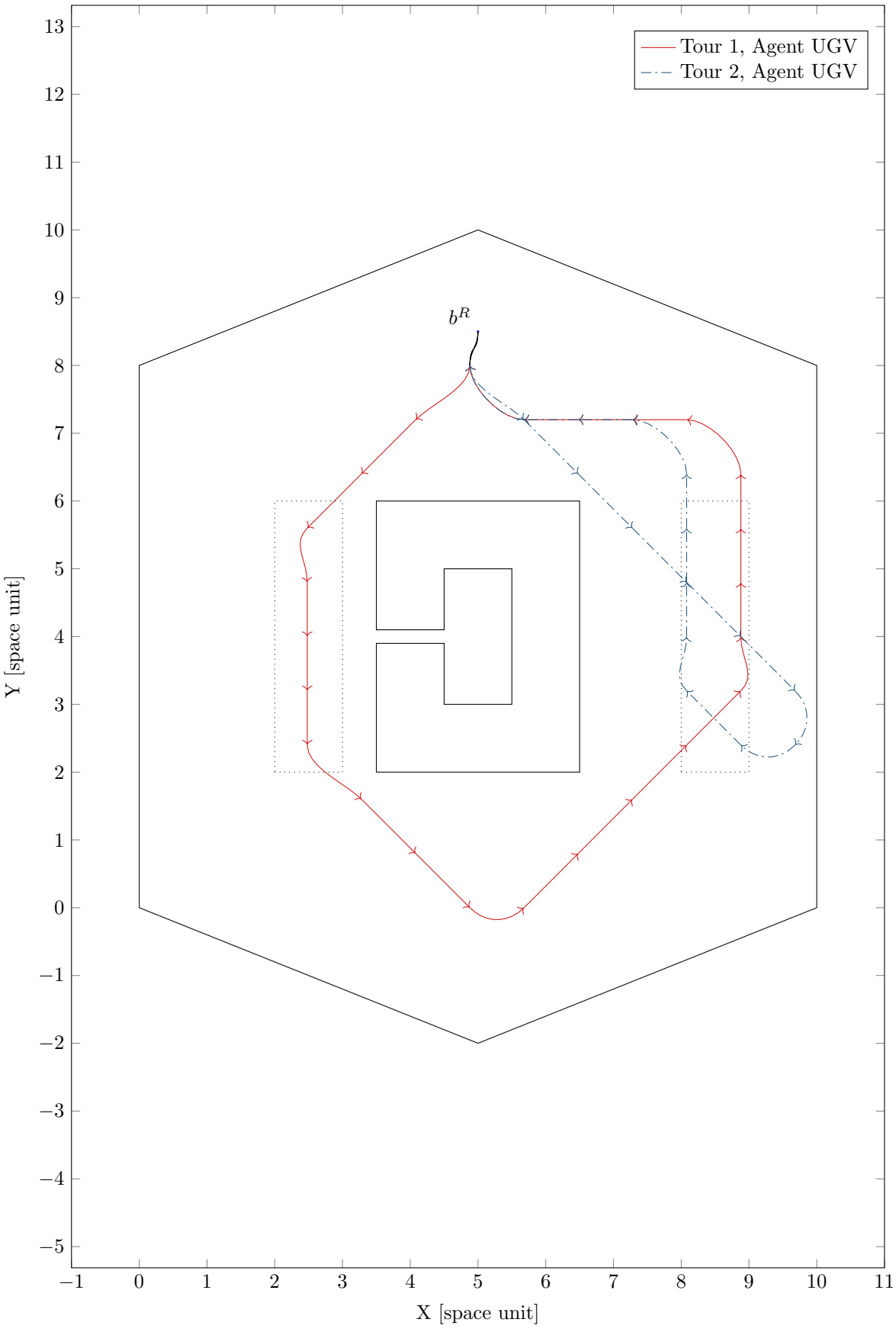

Figure 3.9: Resulting tours for exact solution of case 4.

Table 3.3: Solution Value and timing.

Case	Purpose	Sol. Method	Best Value	Mean Value	Mean Time [s]
1	Steps	ASC+OOS	11.88	–	$1.12 \times 10^{-3} \pm 2.05 \times 10^{-4}$
1	Steps	ASC+ReSC	11.88	–	$1.11 \times 10^{-3} \pm 2.10 \times 10^{-4}$
1	Steps	Exact-Relaxed	11.88	–	$9.00 \times 10^{-3} \pm 1.66 \times 10^{-4}$
2	Collision	ASC+OOS	27.35	–	$1.12 \times 10^{-2} \pm 7.04 \times 10^{-3}$
2	Collision	ASC+ReSC	27.35	–	$1.24 \times 10^{-2} \pm 1.92 \times 10^{-2}$
2	Collision	Exact-Relaxed	27.35	–	$2.69 \times 10^{-2} \pm 4.43 \times 10^{-4}$
3	Cooperation	ASC+OOS	50.48	56.31 ± 2.05	20.20 ± 5.13
3	Cooperation	ASC+ReSC	59.04	66.63 ± 1.32	15.53 ± 6.12
3	Cooperation	Exact-Relaxed	45.12	–	13.60 ± 1.12
4	Distribution	ASC+OOS	33.04	35.71 ± 0.59	4.83 ± 1.28
4	Distribution	ASC+ReSC	33.57	41.83 ± 3.16	3.15 ± 0.76
4	Distribution	Exact-Relaxed	33.04	–	606.44 ± 20.06

one of the heuristics actually achieved the global optimum, having mean suboptimal values standing at about 18% of the global optimum. Hence, it can be seen through these cases that the capacity restriction place the greatest burden on solution techniques for the problem, specially exact general methods, demanding more resources and solution time as the capacity contribution gets more pronounced. ASC's heuristics as proposed here, treats these restrictions naturally via direct tour manipulation, thus retain performance despite tight capacity restrictions. However, the heuristics saw worse performance when dealing with dense regions; a situation that can be remedied by parameters and local optimization algorithm change. Additionally, ASC+ReSC found worse solutions overall, notwithstanding general good results of probabilistic constructive methods in the literature, such as Ant Colony Heuristics. This can be attributed to the ASC itself, since its operations are more akin to ordering than to building, making optimizers like OOS a natural fit. Finally, it becomes clearer that exact methods are more likely to fail in giving any solution at all for bigger instance, since these results were obtained with the relaxed model, which lose representation intrinsically on large-scale situations.

Aside from time and value, some implementations of this methodology may also require memory bounds for the algorithms. Because Julia, which is a Garbage collected language, was selected for this work of prototypical nature, memory estimates may not represent the best possible hardware resource economy, as well as the times obtained may also be reduced by code optimization and native implementation. Nevertheless, by sampling the operational system monitor on some tests, ASC's attained a maximum of 300 Mb memory consumption, while the GUROBI solver attained as much as 3 Gb of memory. It is expected that by implementing the solver on compiled and highly optimized language environments, such as C++ or FORTRAN, the resource usage and heuristic solution time will decrease and increase, respectively, favorably.

These tests, their code and instructions for using them are available at a public web repository made by the authors (JORDAO, 2018) for this work.

4 CONCLUSION AND FUTURE WORKS

A Methodology tackling Persistent Long Term Autonomous Missions for cooperative autonomous vehicles, consisted of heurisitcs able to plan ahead of time all agents' orders, was presented based on a new formalization that reduces the overall problem to a path construction and a combinatorial optimization problem and four case studies were proposed and solved for validation purposes.

The methodology designed is able to successfully produce acceptable continuous routes and tasks for the agents in questions. Moreover, its modular nature enables further studies on other realizations, with different segmentations into topological maps or different solving methodologies with other heuristics and implementations. Some possible improvements follow.

It is within the authors' opinion that the ASC heuristics family is, at least, asymptotically optimal, observing that the bad initial solution can possibly generate every other solution through admissible transformations and its performance can be highly-tuned by change of the local optimization heuristic and scoring system. Confirmation or dismissal of this claim, along with modifications that include seed generation for faster runs is left for future works. Regarding segmentation, a more hierarchical scheme can be used, by shrinking each interest subregion into a single entity with many possible entrances and exists, while the internal cost could be solved with cost transforms commonly used in pure coverage problems. The downside of this change is that the costs of demand coverage in these subregions may rise due to the fixed internal trajectories.

Also, the MILP model may be changed to include only the maximum cost of path operations, instead of their sum; effectively making a single path for same class agents with different tasks, but also requiring a second optimization procedure for dividing said tasks optimally. Additionally, the methodology is already general enough to accommodate non-uniform segmentation schemes, through better use of index sets and broader definition of states for each agent, but then also requiring treatment of sampling schemes as some of the previous works mentioned. For instance, only interest regions can be segmented in a regular basis, whereas the outer non-interest regions can be sampled according to some criteria, say, better SimPaD minimization other than point counting, and then connected via general ShorPaPs, culminating in a hybrid segmentation scheme.

Some remarks can be made about the up and downsides of the methodology realization as presented here. It is apparent that although optimality can be checked and achieved systematically via optimization, if the agents' models change quicker than the mean tour designed by the planner, by environment interaction or hardware degradation, the cost estimate becomes increasingly worse; thus, this realization, as presented, is more suited to missions with slow environmental changes, good a priori knowledge of the mission region and missions that need direct treatment of agents with expressive non-holomonic constraints. Yet, the only major communication required for success of this long term planning is between base and agent, during this agent visit to the operations base, possibly lowering the hardware costs of the mission and also promoting fault tolerance in case of communication breakdown.

Robustness can be achieved by closing an information loop between the agents knowledge of the map through their traversal and the planner current knowledge, as mentioned briefly in Chapter 1. Thus updating the real cost of every tour component after each route is complete, for each agent. Such course is more pertinent to a full integration of the planner with every agent, and is left for future works.

Finally, once these suggestions of improvements are followed, another line of expansion is the use of better models for agents' cost estimation, or even more classes of agents, other than corpuscular entities. Even though this possibility was already covered by the principle of interchangeability of the estimators, the effective change and possible use of more descriptive models, along with real hardware tests, would provide stronger validation for this methodology.

Bibliography

APOSTOL, Tom M. **Calculus, Volume I**. [S.l.]: John Wiley & Sons, 2007. v. 1.

BEKTAŞ, Tolga; GOUVEIA, Luis. Requiem for the Miller-Tucker-Zemlin Subtour Elimination Constraints? **European Journal of Operational Research**, v. 236, n. 3, p. 820–832, 2014. ISSN 03772217. DOI: 10.1016/j.ejor.2013.07.038.

BELLMAN, Richard. On a Routing Problem. **Quarterly of Applied Mathematics**, v. 16, n. 1, p. 87–90, 1958. ISSN 0033-569X, 1552-4485. DOI: 10.1090/qam/102435.

BENTZ, William; HOANG, Tru; BAYASGALAN, Enkhmurun; PANAGOU, Dimitra. Complete 3-D Dynamic Coverage in Energy-Constrained Multi-UAV Sensor Networks. **Autonomous Robots**, p. 1–27, 26 July 2017. ISSN 0929-5593, 1573-7527. DOI: 10.1007/s10514-017-9661-x.

BIRCHER, Andreas; KAMEL, Mina; ALEXIS, Kostas; BURRI, Michael; OETTERSHAGEN, Philipp; OMARI, Sammy; MANTEL, Thomas; SIEGWART, Roland. Three-Dimensional Coverage Path Planning via Viewpoint Resampling and Tour Optimization for Aerial Robots. **Autonomous Robots**, v. 40, n. 6, p. 1059–1078, 1 Aug. 2016. ISSN 0929-5593, 1573-7527. DOI: 10.1007/s10514-015-9517-1.

BOCCALATTE, Marco; BROGI, Filippo; CATALFAMO, Francesco; MADDALUNO, Stefania; MARTINO, Michele; MELLANO, Valter; ROSAZZA PRIN, Paolo; SOLITRO, Filomena; TORASSO, Pietro; TORTA, Gianluca. A Multi-UAS Cooperative Mission Over Non-Segregated Civil Areas. **Journal of Intelligent & Robotic Systems**, v. 70, p. 275–291, 1-4 Apr. 2013. ISSN 0921-0296, 1573-0409. DOI: 10.1007/s10846-012-9706-5.

BRODERICK, John A.; TILBURY, Dawn M.; ATKINS, Ella M. Optimal Coverage Trajectories for a UGV with Tradeoffs for Energy and Time. **Autonomous Robots**, v. 36, n. 3, p. 257–271, 1 Mar. 2014. ISSN 0929-5593, 1573-7527. DOI: 10.1007/s10514-013-9348-x.

BURGER, Mernout; HUISKAMP, Marco; KEVICZKY, Tamás. Complete Field Coverage as a Multi-Vehicle Routing Problem. **IFAC Proceedings Volumes**, v. 46, n. 18, p. 97–102, Aug. 2013. ISSN 14746670. DOI: 10.3182/20130828-2-SF-3019.00050.

CAMPOS, V.; MOTA, E. Heuristic Procedures for the Capacitated Vehicle Routing Problem. **Computational Optimization and Applications**, v. 16, n. 3, p. 265–277, Sept. 2000. ISSN 0926-6003, 1573-2894. DOI: 10.1023/A:1008768313174.

CORMEN, Thomas H. **Introduction to Algorithms**. [S.l.]: MIT press, 2009.

DAMILANO, Luca; GUGLIERI, Giorgio; QUAGLIOTTI, Fulvia; SALE, Ilaria; LUNGHI, Alessio. Ground Control Station Embedded Mission Planning for UAS. **Journal of Intelligent & Robotic Systems**, v. 69, p. 241–256, 1-4 Jan. 2013. ISSN 0921-0296, 1573-0409. DOI: 10.1007/s10846-012-9697-2.

DESROCHERS, Martin; LAPORTE, Gilbert. Improvements and Extensions to the Miller-Tucker-Zemlin Subtour Elimination Constraints. **Operations Research Letters**, v. 10, n. 1, p. 27–36, 1991.

DIJKSTRA, E. W. A Note on Two Problems in Connexion with Graphs. **Numerische Mathematik**, v. 1, n. 1, p. 269–271, Dec. 1959. ISSN 0029-599X, 0945-3245. DOI: 10.1007/BF01386390.

DORIGO, Marco; BLUM, Christian. Ant Colony Optimization Theory: A Survey. **Theoretical Computer Science**, v. 344, n. 2, p. 243–278, 17 Nov. 2005. ISSN 0304-3975. DOI: 10.1016/j.tcs.2005.05.020.

ELFES, ALBERTO; BUENO, Samuel S.; BERGERMAN, Marcel; RAMOS, Josua Jr. G.; GOMES, Sao Bittencourt Varella. Project AURORA: Development of an Autonomous Unmanned Remote Monitoring Robotic Airship. **Journal of the Brazilian Computer Society**, v. 4, Apr. 1998. ISSN 0104-6500.

FAZLI, Pooyan; DAVOODI, Alireza; MACKWORTH, Alan K. Multi-Robot Repeated Area Coverage. **Autonomous Robots**, v. 34, n. 4, p. 251–276, 1 May 2013. ISSN 0929-5593, 1573-7527. DOI: 10.1007/s10514-012-9319-7.

FELNER, Ariel. Position Paper: Dijkstra's Algorithm versus Uniform Cost Search or a Case against Dijkstra's Algorithm. In: FOURTH Annual Symposium on Combinatorial Search. [S.l.: s.n.], 2011.

FISHER, Marshall L.; JAIKUMAR, Ramchandran. A Generalized Assignment Heuristic for Vehicle Routing. **Networks**, v. 11, n. 2, p. 109–124, 1981. ISSN 10970037. DOI: 10.1002/net.3230110205.

GLOVER, Fred. Future Paths for Integer Programming and Links to Artificial Intelligence. **Computers & operations research**, v. 13, n. 5, p. 533–549, 1986.

GOLDEN, Bruce; RAGHAVAN, S.; WASIL, Edward. **The Vehicle Routing Problem: Latest Advances and New Challenges**. [S.l.: s.n.], 2008. v. 43. 589 pp. ISBN 978-0-387-77777-1. DOI: 10.1007/978-0-387-77778-8.

GUROBI OPTIMIZATION, Inc. **Gurobi Optimizer Reference Manual**. [S.l.: s.n.], 2016.

INPE. **PRODES**. Sao Jose dos Campos, 2017.

ISAACS, Jason T.; HESPANHA, Joao P. Dubins Traveling Salesman Problem with Neighborhoods: A Graph-Based Approach. **Algorithms**, v. 6, n. 1, p. 84–99, 2013. ISSN 19994893. DOI: 10.3390/a6010084.

JAVANMARD ALITAPPEH, Reza; A. S. PEREIRA, Guilherme; R. ARAÚJO, Arthur; C. A. PIMENTA, Luciano. Multi-Robot Deployment Using Topological Maps. **Journal of Intelligent and Robotic Systems: Theory and Applications**, v. 86, p. 641–661, 3-4 2017. ISSN 15730409. DOI: 10.1007/s10846-017-0471-3.

JORDAO, Rodolfo. **JordaoRodolfo / PLTAM-2017-Paper-Companion / Admin / Repository Details — Bitbucket**. Available from: ¡https://bitbucket.org/JordaoRodolfo/pltam-2017-paper-companion/admin¿. Visited on: 24 Jan. 2018.

KARAMAN, Sertac; FRAZZOLI, Emilio. Sampling-Based Algorithms for Optimal Motion Planning, p. 1–76, 2011. ISSN 0278-3649. DOI: 10.1177/0278364911406761.

KIRK, Donald E. **Optimal Control Theory: An Introduction**. Mineola, N.Y: Dover Publications, 2004. 452 pp. ISBN 978-0-486-43484-1.

LANGEVIN, André; SOUMIS, François; DESROSIERS, Jacques. Classification of Travelling Salesman Problem Formulations. **Operations Research Letters**, v. 9, n. 2, p. 127–132, 1990.

LAVALLE, Steven M. **Planning Algorithms**. [S.l.]: Cambridge university press, 2006.

MITCHELL, Derek; CORAH, Micah; CHAKRABORTY, Nilanjan; SYCARA, Katia; MICHAEL, Nathan. Multi-Robot Long-Term Persistent Coverage with Fuel Constrained Robots. In: PROCEEDINGS - IEEE International Conference on Robotics and Automation. [S.l.: s.n.], 2015. 2015-June, p. 1093–1099. ISBN 978-1-4799-6923-4. DOI: 10.1109/ICRA.2015.7139312.

NASA, JPL. **Mars Science Laboratory**. Available from: ¡https://mars.nasa.gov/msl/¿. Visited on: 5 Dec. 2017.

PADEN, Brian; CAP, Michal; YONG, Sze Zheng; YERSHOV, Dmitry; FRAZZOLI, Emilio. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles, p. 1–27, 2016. ISSN 2379-8904. DOI: 10.1109/TIV.2016.2578706.

PINAGÉ, Felipe A.; CAVALHO, José Reginaldo H.; QUEIROZ-NETO, José P. Visual-Based Natural Landmark Tracking Method to Support UAV Navigation over Rain Forest Areas. In: VISAPP. [S.l.: s.n.], 2013.

PIVTORAIKO, Mikhail; KNEPPER, Ross Alan; KELLY, Alonzo. Differentially Constrained Mobile Robot Motion Planning in State Lattices. **Journal of Field Robotics**, v. 26, n. 3, p. 308–333, Mar. 2009.

SICILIANO, Bruno. **Robotics : Modelling, Planning and Control**. London: Springer, 2009. ISBN 978-1-84628-642-1.

TOTH, Paolo; VIGO, Daniele. **The Vehicle Routing Problem**. [S.l.: s.n.], 2002. v. 9. ISBN 0-89871-579-2.

TSOURDOS, Antonios; WHITE, Brian; SHANMUGAVEL, Madhavan. **Cooperative Path Planning of Unmanned Aerial Vehicles**. [S.l.: s.n.], 2010.

VIDAL, Thibaut; CRAINIC, Teodor Gabriel; GENDREAU, Michel; PRINS, Christian. A Unified Solution Framework for Multi-Attribute Vehicle Routing Problems. **European Journal of Operational Research**, v. 234, n. 3, p. 658–673, 2014. ISSN 03772217. DOI: 10.1016/j.ejor.2013.09.045.

_____. Heuristics for Multi-Attribute Vehicle Routing Problems: A Survey and Synthesis. **European Journal of Operational Research**, v. 231, n. 1, p. 1–21, 2013. ISSN 03772217. DOI: 10.1016/j.ejor.2013.02.053.

WOEGINGER, Gerhard. **The P-versus-NP Page**. [S.l.: s.n.], 27 Feb. 2018.

XU, Anqi; VIRIYASUTHEE, Chatavut; REKLEITIS, Ioannis. Efficient Complete Coverage of a Known Arbitrary Environment with Applications to Aerial Operations. **Autonomous Robots**, v. 36, n. 4, p. 365–381, 1 Apr. 2014. ISSN 0929-5593, 1573-7527. DOI: 10.1007/s10514-013-9364-x.

ZAGER, Laura A.; VERGHESE, George C. Graph Similarity Scoring and Matching. **Applied Mathematics Letters**, v. 21, n. 1, p. 86–94, 2008. ISSN 08939659. DOI: 10.1016/j.aml.2007.01.006.