Connected Multi-Agent Path Finding: How Robots Get Away with Texting and Driving

Arthur Queffelec

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

THÈSE DE DOCTORAT DE

L'UNIVERSITÉ DE RENNES 1

ÉCOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Par
Arthur QUEFFELEC

Connected Multi-Agent Path Finding

How Robots Get Away with Texting and Driving

Thèse présentée et soutenue à Rennes, le 11 Octobre 2021
Unité de recherche : IRISA - UMR6074
Thèse N° :

Rapporteurs avant soutenance :

Sven Koenig Professeur, University of Southern California
Bruno Zanuttini Professeur, Université de Caen

Composition du Jury :

Attention, en cas d'absence d'un des membres du Jury le jour de la soutenance, la composition du jury doit être revue pour s'assurer qu'elle est conforme et devra être répercutée sur la couverture de thèse

Président :	Nicolas Markey	Directeur de recherche, IRISA, Inria, CNRS, Univ. Rennes
Examinateurs :	Sven Koenig	Professeur, University of Southern California
	Anastasia Paparrizou	Chargée de recherche, Université d'Artois, CRIL-CNRS
	Bruno Zanuttini	Professeur, GREYC, Normandie Université, UNICAEN, CNRS, ENSICAEN
Dir. de thèse :	François Schwarzentruber	Maitre de conférences, IRISA, CNRS, Univ. Rennes
Encadrant :	Ocan Sankur	Chargé de recherche, IRISA, Inria, CNRS, Univ. Rennes

RESUMÉ

La planification de chemin consiste à concevoir une séquence d'étapes à suivre pour une entité mobile. Cette tâche est au cœur de nombreux problèmes du monde réel. La planification est utilisée dans les problèmes d'optimisation pour les trains, dans l'aviation et même pour analyser/concevoir les intersections routières. En outre, l'utilisation récente de véhicules autonomes a accru le besoin de planification, notamment pour les robots d'entrepôt, les véhicules de remorquage d'avions, les robots de bureau, les véhicules portuaires autonomes, ou même pour les entités virtuelles de jeux vidéo. De plus, la planification peut garantir de nombreuses propriétés comme l'optimalité, l'absence de collisions, la connectivité périodique/continue ou la préservation d'une formation. L'étude de la planification autonome peut permettre de réduire la congestion, la pollution, les accidents, les coûts et plus encore.

Dans certaines applications, il est important de considérer la connectivité des agents. Bien que certaines configurations garantissent une connectivité permanente entre les entités (ex : entrepôts), ce n'est pas toujours le cas dans les applications avec des environnements ouverts. Prenons l'exemple des missions de collecte d'informations, au cours desquelles les agents doivent visiter un ensemble de lieux et recueillir des informations à l'aide de capteurs (e.g. des caméras, des capteurs de fumée, des hygromètres, etc.) Une exigence commune de ces missions est d'obtenir un flux vidéo en direct, ce qui ne peut être assuré que par une connectivité permanente. Cette exigence peut être critique dans des contextes spécifiques tels que les missions de recherche et de sauvetage, où la communication entre les agents dépend de la topologie de la zone (e.g. les bâtiments, les arbres) et où elle est nécessaire pour assurer une visite appropriée de la zone, ou pour transférer les informations recueillies à une station de base.

Un autre aspect que l'on retrouve dans de nombreuses applications est le manque de la connaissance complète de la zone dans laquelle les entités se déplacent. Par exemple, dans les missions d'exploration, les agents ne reçoivent aucune information sur l'environnement et doivent le découvrir par eux-mêmes. Dans ce contexte, les agents peuvent être amenés à assurer une connectivité à tout moment pour échanger leurs résultats et, en cas de déconnexion, ils devront se rechercher pour se reconnecter.

Les travaux récents sur la planification des entités se sont concentrés sur la planification d'exécutions sans collision. Ce problème, appelé Multi-Agent Path Finding (MAPF), consiste à trouver une séquence d'étapes pour qu'un groupe d'agents atteigne des cibles spécifiées tout en évitant les collisions. Cette thèse étudie deux aspects de la planification de chemins multi-agents afin de modéliser et de résoudre des applications plus réalistes : (1) la connectivité généralisée des agents ; (2) la connaissance incomplète de l'environnement.

De nombreux travaux sur la planification des chemins et la planification de couvertures supposent un modèle de graphe discret donné obtenu par décomposition cellulaire ou graphe de visibilité, ou utilisent une méthode d'échantillonnage pour construire un graphe sur lequel des algorithmes combinatoires sont appliqués. Les algorithmes basés sur les graphes, tels que ceux que nous étudions, et les travaux antérieurs tels que A* pour MAPF sont donc pertinents dans

ce contexte.

Il est important de comprendre la complexité computationnelle de ces problèmes de graphes pour connaître les limites des solutions algorithmiques ainsi que les heuristiques qui peuvent être appliquées au problème en question. La contribution de ce travail est double. Tout d'abord, nous présentons un cadre pour étudier et modéliser les problèmes de planification de chemins multi-agents basés sur la connectivité. Nous fournissons un travail initial détaillé sur la complexité de ce cadre et un algorithme optimal pour le résoudre. Deuxièmement, nous étendons notre cadre de connectivité au cadre de connaissance incomplète et montrons la complexité du calcul connecté et décentralisé des plans dans des environnements partiellement connus.

Tout d'abord, nous abordons le problème de la connectivité, appelé Connected Multi-Agent Path Finding (MAPF). Bien que des travaux initiaux aient été effectués pour formaliser un cadre de communication basé sur l'environnement, nous avons étudié ce problème en détail sous deux objectifs différents, l'accessibilité et la couverture. Le premier problème consiste à rechercher une séquence d'étapes pour qu'un groupe d'agents navigue dans un environnement pour atteindre un emplacement spécifique. Le second problème consiste à trouver une exécution pour une flotte afin de couvrir tous les emplacements d'une zone. Un aspect important de ces problèmes est de savoir si on nous donne une limite sur le nombre d'étapes dans lesquelles les agents doivent atteindre leurs objectifs. Il est intéressant de noter que ces problèmes se comportent différemment selon la classe d'instance. En particulier, nous montrons la difficulté de ces problèmes en général et les restrictions possibles qui les rendent traitables. En outre, nous présentons la complexité de certaines variantes simples qui peuvent être déduites des preuves. Nos résultats montrent que les problèmes de recherche de chemin ou de couverture sont intraitables que l'exécution soit bornée ou non, NP-complet et PSPACE-complet respectivement. Cependant, décider de l'existence d'un plan peut devenir plus facile sous certaines hypothèses, surtout si l'on se restreint aux graphes sight-moveable. En effet, avec cette classe, qui restreint la communication aux emplacements assurant un chemin connecté, les problèmes sont LOGSPACE ou NLOGSPACE. Cependant, nous avons montré que l'optimisation d'une exécution est un problème difficile à moins de se restreindre à des classes irréalistes.

Ensuite, nous illustrons et analysons un algorithme optimal pour résoudre CMAPF. Certains algorithmes ont été proposés précédemment : un algorithme en ligne et deux algorithmes d'approximation. Contrairement aux approches précédentes, la structure de notre algorithme est basée sur CBS, et nous prouvons son optimalité et sa correction. L'approche est découplée, un algorithme de bas niveau recherche un chemin pour chaque agent, tandis qu'un algorithme de haut niveau contraint les agents pour assurer la connectivité. Nous discutons de certaines optimisations, et nous fournissons une étude expérimentale de son passage à l'échelle.

Nous présentons un cadre pour la connaissance incomplète prenant en compte la connectivité. Nous utilisons le cadre de communication de la première contribution, et limitons notre étude au problème de l'accessibilité. La connaissance incomplète de l'environnement fait référence au fait que certains mouvements ou communications ne sont pas assurés. Ainsi, même si l'environnement est statique, les agents vont observer de nouvelles informations lors de leur exploration. Dans ce cas, il est important de distinguer deux problèmes. Premièrement, le cas connecté qui impose aux agents de rester connectés à toutes les étapes. Dans ce cas, les agents doivent s'assurer qu'une communication incertaine ne perturbera pas l'exécution. Cependant,

les agents partagent la même connaissance à tout moment puisqu'ils s'informent mutuellement de leurs résultats. Le deuxième problème, le cas décentralisé, laisse les agents se déconnecter pendant la mission. Comme les agents ne peuvent pas partager leurs connaissances à tout moment, ils doivent raisonner sur les observations possibles des autres et leurs choix.

Cette différence est un indice de l'explication du "saut" conséquent en complexité entre le cas connecté et le cas décentralisé. Nos résultats montrent que la distinction intuitive entre le cas décentralisé et le cas connecté a un impact majeur sur la complexité du problème. En effet, le cas connecté est PSPACE-complet et le cas décentralisé est NEXPTIME-complet. Dans le cadre général des graphes dirigés, l'ajout d'une borne sur l'exécution n'a aucun impact sur la complexité. Cependant, sur les graphes non orientés, il est trivial de trouver une exécution non bornée, car les agents peuvent explorer toutes les configurations jusqu'à ce qu'ils atteignent le but ou découvrent qu'il est impossible de le faire.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Path planning is the task of designing a sequence of steps for a mobile entity to follow. This task is required at the center of numerous real-world problems. Planning is used in optimization problems for trains [56], in aviation [119] and even in the analysis/design of road intersections [39]. In addition, the recent use of autonomous vehicles has increased the need for planning, namely for warehouse robots [162], aircraft-towing vehicles [110], office robots [157], autonomous port vehicles [154], or even for virtual entities as in video-games [106]. Additionally, planning of agents can guarantee many properties as optimality and lack of collisions [139], periodic/continuous connectivity [69] or preservation of a formation [93]. The study of autonomous planning can allow one to reduce congestion, pollution, accidents, costs and more.

In some applications, it is important to consider the *connectivity* of the agents. Although some settings guarantee a permanent connectivity among entities (e.g. warehouses [162]), this is not always true in applications with open environments. As an example, consider information gathering missions, where agents are required to visit a set of locations and gather information using sensors (e.g. cameras, smoke sensors, hygrometers etc.). A common requirement of these missions is to obtain a live feed of video which can be ensured only through permanent connectivity [73]. This requirement can be critical in specific setting like search and rescue missions, where the communication between the agents depends on the topology of the area (e.g. buildings, trees) and it is required to ensure proper visit of the area, or to transfer information gathered to a base station [3, 151].

Another aspect that can be found in many applications is the lack of complete knowledge of the area in which the entities move. For instance, in exploration missions, the agents are not provided any information of the environment and must discover it by themselves [127, 153]. In this context, the agents may have to ensure connectivity at all times to exchange their findings and in case of disconnection they will have to search for each other to reconnect.

Recent works on the planning of entities have been focused on the planning of collision-free executions. This problem, called Multi-Agent Path Finding (MAPF), is to find a sequence of steps for a group of agents to reach specified targets while avoiding collisions. This dissertation studies two aspects of multi-agent path planning in order to model and solve more realistic applications: (1) generalized connectivity of the agents; (2) incomplete knowledge of the environment.

Many works on path planning and coverage path planning either assume a given discrete graph model obtained by cell decomposition or visibility graph [87], or use a sampling method to construct a graph on which combinatorial algorithms are applied [79, 83]. Graph-based algorithms such as those we study, and previous work such as A^* for MAPF [133, 145] and Conflict-Based Search (CBS) [137] are thus relevant in this setting. It is important to under-

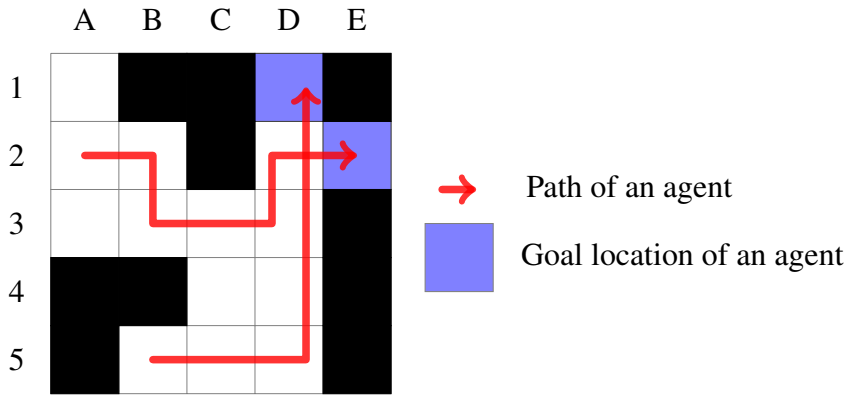


Figure 1.1 – Example of an execution for two agents in a 2D environment. The start locations are A2 and B5. The goal locations are E2 and D1.

stand the computational complexity of these graph problems to understand the limits of the algorithmic solutions and as well as heuristics that can be applied to the problem at hand.

In Figure 1.1, we depicted a 2D environment and an execution of two agents. The environment is represented by black and white tiles. The former represents obstacles and the latter represents free areas. An agent can move to either of the white tiles surrounding his current position (i.e. North, South, East, West). In this example the two agents are in collision after 4 steps in the cell D3. This execution is, thus, not collision-free and would not be valid for the MAPF problem. A solution would be to delay one of the agent by 1 step.

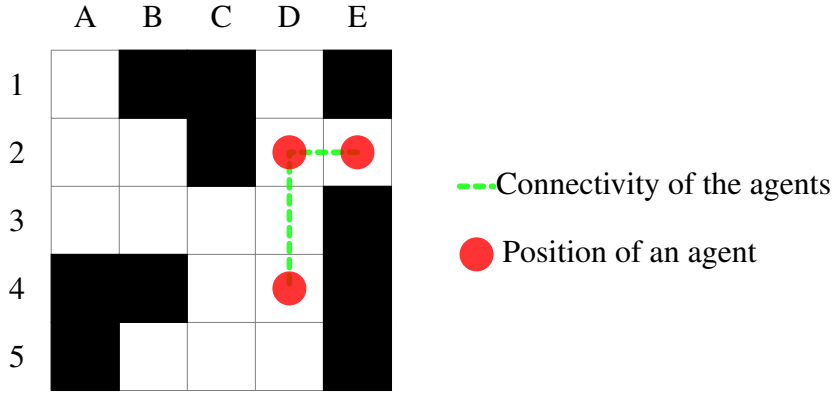


Figure 1.2 – Example of a connected configuration for three agents on a 2D environment.

Figure 1.2 depicts a connected configuration of three agents. It is worth noting that agents are capable of *multi-hop* communication. In other words, the agent can *relay* communication to others. This phenomenon is depicted by agent in D4 connected to agent in E2 through the agent in D2.

An example of a partially-known environment is depicted in Figure 1.3. We consider that the agent can see at 2 cells away from them (only 1 cell in diagonal). In order to reach their goals, the agents must avoid the obstacles that they will encounter during the mission. It is

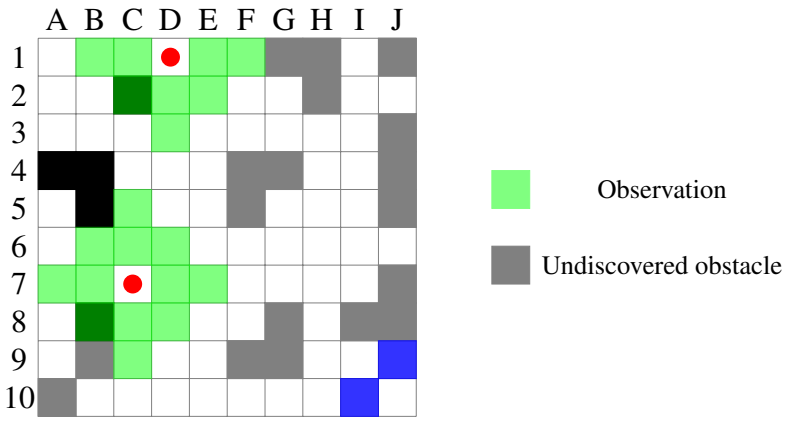


Figure 1.3 – Example of a state in the exploration of a partially-known environment with 2 agents.

worth noting that it may be beneficial for the agents to take a detour to observe the environment as it could shorten the paths of others. Indeed, it may be the case that an agent by, extending slightly its path and, observing the presence of an obstacle, can warn in advance another agent, shortening the overall execution.

1.1 Contribution

The contribution of this work is twofold. First, we present a framework to study and model connectivity-based multi-agent path planning problems. We provide a detailed initial work on the complexity of this framework and an optimal algorithm to solve it. Second, we extend our connectivity framework to the incomplete knowledge setting and show the complexity of the connected and decentralized computation of plans under partially known environments. Let us give a more detailed summary of our contributions.

1.1.1 Connectivity

Firstly, we address the problem of connectivity. While some initial work has been done to formalize an *environment-based* communication framework [69, 151], we studied this problem in detail under two different goals, reachability and coverage. The summary of the contributions is depicted in Table 1.1.

The reachability problem is denoted **CMAPF**, for Connected Multi-Agent Path Finding, and the coverage problem is denoted **CMACP**, for Connected Multi-Agent Coverage Planning. The first problem consists in searching a sequence of steps for a group of agents to navigate in an environment to reach a specific location. The **CMACP** problem asks to find an execution for a fleet to cover every location of an area.

An important aspect of those problems is whether we are given a bound on the number of steps in which the agents are to achieve their goals. We will denote the *bounded* version of

a problem with the prefix 'b'. For instance, **bCMACP** is the problem of finding a connected bounded covering execution.

It is worth noting that these problems behave differently depending on the class of instance. In particular, we show how hard these problems are in general and possible restrictions which render them tractable. In addition, we present the complexity of some simple variants that can be inferred from the proofs.

Our results show that path finding or coverage problems are intractable whether the execution is bounded or not, NP-complete and PSPACE-complete respectively. However, deciding the existence of a plan can become easier under some assumptions, especially if restricted to Sight-Moveable graphs. Indeed, with this class, which restrict the communication to locations ensuring a connected path, the problems are LOGSPACE or NLOGSPACE. However, we showed that the optimization of an execution is a hard problem unless restricted to unrealistic classes.

Then, we illustrate and analyze an optimal algorithm to solve Connected Multi-Agent Path Finding (CMAPF). Some algorithms have been previously proposed: an online algorithm [68] and two approximation algorithms [151]. Contrarily to the previous approaches, the structure of our algorithm is based on CBS, and we prove its optimality and soundness. The approach is decoupled, a low-level algorithm searches for a path for each agent, while a high-level algorithm constrains the agents to ensure connectivity. We discuss some optimizations, and we provide an experimental study of its scalability.

1.1.2 Incomplete Knowledge

We present a setting for incomplete knowledge taking connectivity into account. We use the communication framework of the first contribution, and restrict our study to the reachability problem. A problem with a suffix 'I'denotes the incomplete knowledge variant. For instance, **bCMAPFI** denotes the bounded reachability problem with incomplete knowledge. The summary of the contributions is depicted in Table 1.2.

The incomplete knowledge of the environment refers to the fact that some movement or communication are not ensured. Thus, even though the environment is static the agents will observe new information as they *explore*. In this case, it is important to distinguish two problems. First, the *connected* case which enforces the agents to stay connected at all steps. In this case, the agents have to ensure that an uncertain communication will not disrupt the execution. However, the agents share the same knowledge at all times as they inform each other of their findings. The second problem, the *decentralized* case, lets the agents disconnect during the mission. As the agents may not share their knowledge at all times, they have to reason about the possible observations of the others and their choices.

This difference is a hint at the explanation to the consequent "jump" in complexity between the connected case, **CMAPFI**, and the decentralized case, **DMAPFI**. Our results show that the intuitive distinction between the decentralized case and the connected case has a major impact on the complexity of the problem. Indeed, the connected case is PSPACE-complete and the decentralized case is NEXPTIME-complete. In the general setting of directed graphs, the addition of a bound on the execution has no impact on the complexity. However, on undirected

graphs it is trivial to find an unbounded execution as the agents can explore all configurations until they reach the goal or discover that it cannot be done.

1.2 Outline

The rest of this dissertation is structured in 6 Chapters as follows:

In Chapter 2, we present the preliminaries. In particular, the Section 2.1 introduces the necessary notions of complexity theory and Section 2.2 formalizes MAPF. The Chapter 3 summarizes the state of the art. More precisely, in Section 3.1 we give a background of MAPF, Section 3.2 is dedicated to present previous works on coverage planning, in Section 3.3 we introduce some work on connectivity and Section 3.4 presents the work which inspired our incomplete knowledge framework. Then, the contributions of this dissertation are arranged in two parts, with Part I dedicated to the study of CMAPF, and Part II focused on the study of an incomplete knowledge setting for the coordination of agents.

In Chapter 4, we introduce and study the connectivity framework. Section 4.1 establishes the setting of our work. In Section 4.2, we introduce the required notions for the rest of the chapter. In Section 4.3, we describe the upper bounds of our problems on directed topological graphs and, in Section 4.4, we prove the lower bounds on undirected topological graphs to obtain completeness results. In Section 4.5, we study sight-moveable topological graphs. Section 4.6 contains the complexity analysis of the complete-communication topological graphs. We introduce relevant extensions of our problems in Section 4.7. We present a conclusion and future works in Section 4.8.

Chapter 5 introduces an algorithm to solve CMAPF. Section 5.1 gives an overview of the setting and previous works. Section 5.2 presents our algorithm, and Section 5.3 contains the correctness and optimality proofs. Then, Section 5.4 presents experiments.

In Chapter 6, we study the problem of connected coordination in a partially-known environment. Section 6.1 presents the study of incomplete knowledge. In Section 6.2, we formalize our imperfect information setting. We recall the TQBF and TDQBF problems in Section 6.3. Our results for the connected and decentralized cases are given in Sections 6.4 and 6.5 respectively. Section 6.6 contains additional results that can be obtained from our work, discussion on related work, and perspectives.

Finally, in Chapter 7, we conclude on the work achieved in this dissertation. We summarize the different results of each contribution in Section 7.1. In Section 7.2, we provide the list of articles that have been published during the writing of this study. Then, we state possible future works that would extend our results and remove limitations in Section 7.3.

<table>
<tr><th>Top. Graph/Problem</th><th>CMAPF
(Def. 4.7)</th><th>CMACP
(Def. 4.8)</th><th>bCMAPF
(Def. 4.9)</th><th>bCMACP
(Def. 4.10)</th></tr>
<tr><td>Directed
(Def. 4.1)</td><td rowspan="2">PSPACE-complete
(Th. 4.3)</td><td rowspan="2">PSPACE-complete
(Th. 4.4)</td><td rowspan="2">NP-complete
[69]</td><td rowspan="4">NP-complete
(Th. 4.8)</td></tr>
<tr><td>Undirected
(Def. 4.2)</td></tr>
<tr><td>Sight-Moveable
(Def. 4.3)</td><td rowspan="2">in LOGSPACE
(Prop. 4.3)</td><td rowspan="2">in NLOGSPACE
(Prop. 4.4)</td><td>NP-complete
(Th. 4.6)</td></tr>
<tr><td>Complete-Comm.
(Def. 4.4)</td><td>in NLOGSPACE
(Prop. 4.5)</td></tr>
</table>

Table 1.1 – Summary of the contribution (1).

<table>
<tr><th>Top. Graph/Problem</th><th>CMAPFI
(Def. 6.4)</th><th>DMAPFI
(Def. 6.3)</th><th>bCMAPFI
(Def. 6.2)</th><th>bDMAPFI
(Def. 6.1)</th></tr>
<tr><td>Directed
(Def. 4.1)</td><td>PSPACE-complete
(Th. 6.1)</td><td>NEXPTIME-complete
(Th. 6.4)</td><td rowspan="2">PSPACE-complete
(Th. 6.2, 6.3)</td><td rowspan="2">NEXPTIME-complete
(Th. 6.5)</td></tr>
<tr><td>Undirected
(Def. 4.2)</td><td colspan="2">Trivial
See 6.6.1</td></tr>
</table>

Table 1.2 – Summary of the contribution (2).

CHAPTER 2

PRELIMINARIES

This chapter is dedicated to the introduction of the concepts that require a proper definition for the results presented in this dissertation. We first introduce notions from complexity theory. This theory will be used throughout this work as we will classify the difficulty of several studied problems. This classification work has for primary purpose to evaluate how efficiently a problem can be solved. In addition, the complexity proofs provide an insight on the difficulties of a problem.

Then, we present the MAPF problem formally. This problem is at the root of this work and the definition will be extended to account for new aspects such as connectivity and incomplete knowledge.

2.1 Complexity Theory

Complexity theory is the study of algorithmic problems and how hard they are to solve. Let us begin with the notion of *decision problem*, *complexity class* and *completeness*. The following section is inspired from Garey and Johnson [55]. See the books of Garey and Johnson [55], Lewis and Papadimitriou [89], Arora and Barak [7], and Homer and Selman [70] for a detailed presentation of the complexity theory.

2.1.1 Problem

In this work, we consider a decision problem to be a task to accomplish. A task takes place in a specific *domain* and can either be completed or not. As an example, let us suppose that we want to deliver a package. For this delivery, we are given the *instance*: a map of the city, our starting location, and the delivery location. First and foremost, we must make sure that we can reach the delivery location. In other words, we must *decide* whether our task can be completed, if the current instance is *positive* or not. More formally, we define a decision problem as follows.

Definition 2.1 (Decision Problem)**.** *A decision problem is a pair* $\Pi = (D, Y)$ *with* D*, the domain, a set of instances and* $Y \subseteq D$ *a set of positive instances.*

From now on, when defining a problem, the *input* — describing the generic information provided — and the *output* — stating a question over the input — represent the domain and the positive instances, respectively. Let us define the Undirected Connectivity problem to illustrate the previous definition.

Problem (Undirected Connectivity)**.**

- Input: *A graph $G = \langle V, E \rangle$, with V a finite set of vertices and E a finite set of undirected edges, and two vertices $s, t \in V$.*
- Output: *Does there exist a path from s to t?*

One can wish to ask a maximization or minimization question over the output. For instance, we can assume that our previous delivery problem is requiring to be done as fast as possible. This new task is an *optimization problem* which asks to minimize our path from s to t. It is worth noting that, a decision problem can be derived from an optimization problem. Indeed, by the introduction of a bound k, one can reformulate the yes-no question.

Problem (Undirected Shortest Path)**.**

- Input: *A graph $G = \langle V, E \rangle$, with V a finite set of vertices and E a finite set of undirected edges, two vertices $s, t \in V$, and a bound k.*
- Output: *Does there exist a path from s to t of cost no more than k?*

Under the assumption that the cost is "easy" to compute, this decision problem is not harder than the associated optimization problem. Indeed, one can solve the optimization problem and compare the cost of the obtained solution with the given bound. Thus, the associated optimization problem is at least as hard.

Encoding

While the presentation of the formal encoding of a decision problem is beyond of the scope of this dissertation, a detail is worth bringing to the attention of the reader. When deriving a decision problem from an optimization problem, the exact encoding of the bound may be important to specify. Indeed, the complexity of a problem may increase significantly with the use of a more succinct encoding.

In this work, we will often demonstrate the complexity of a problem using a binary encoding of the bound, as one might call it more "reasonable". However, when relevant, we show whether the complexity holds or changes using a unary encoding for completeness.

Deterministic Turing Machine

To define the notion of *algorithm*, we must first present the model of computation. For simplicity, we model an algorithm by the two-tapes Deterministic Turing Machine (DTM), which consists of two *infinite tapes* (input, work), a *finite control* with two *heads*. A *read-only* head is attached to the input tape and a *read/write* head is attached to the work tape.

The tapes of the machine are composed of linearly ordered cells, each holding a symbol. The control is composed of a finite set of *states*, including the distinguished *start*, *yes* and *no* states. A depiction of a DTM is provided in Figure 2.1.

With the help of the head, the control can read the content of a cell from the input and work tapes, or write only to the work tape. Additionally, it can move the head to the left or right,

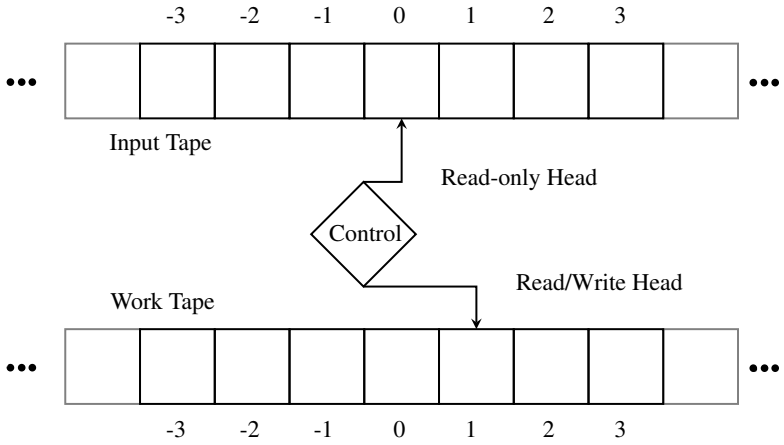


Figure 2.1 – Schematic representation of a two-tape DTM.

and change state. The machine starts with the control in its start state and the heads on the cells numbered 1 of their respective tape. From there, the steps taken by the control is determined by its *internal state* and the content of the cell being read. When the control reaches the yes or no state, it halts and "answers" accordingly.

Now, let us assume that given a problem Π, we have a DTM which, for all instances $d \in D$ in the domain, answers "yes" if $d \in Y$ and "no" if $d \notin Y$. Then we say that this DTM is a *deterministic algorithm* that *solves* the problem Π.

2.1.2 Complexity Classes

Time

In order to solve a problem, an algorithm must go through a number of steps during its computation. This idea can be related to the *time* used by the algorithm to halt. More formally, we define the time function as follows.

Definition 2.2 (Time Function)**.**

$$\mathcal{T}_{\mathcal{A},\Pi}(n) = \max_{d \in D, |d|=n} \{m \mid \mathcal{A} \text{ solves } d \text{ in } m \text{ steps } \}$$

is the most amount of time which the algorithm $\mathcal{A}$ uses to solve an instance of size n of a problem Π.

We can say that $\mathcal{A}$ is a *polynomial-time* algorithm if there exists a polynomial p such that, for all $n \in \mathbb{Z}^+, \mathcal{T}_{\mathcal{A},\Pi}(n) \ll p(n)$. Equivalently, we can say that $\mathcal{A}$ is an *exponential-time* algorithm if there exists an exponential e such that, for all $n \in \mathbb{Z}^+, \mathcal{T}_{\mathcal{A},\Pi}(n) \ll e(n)$.

We can now derive time-classes of problems:

Definition 2.3 (PTIME)**.** *The class PTIME, or P for short, is the class of problems for which there exists a polynomial-time algorithm which solves it.*

Definition 2.4 (EXPTIME)**.** *The class EXPTIME is the class of problems for which there exists an exponential-time algorithm which solves it.*

Space

During computation, an algorithm not only takes steps but uses space on the work tape of the DTM to store information. This can be called the "memory" of the algorithm. More formally, we define the space function as follows.

Definition 2.5 (Space Function)**.**

$$\mathcal{S}_{\mathcal{A},\Pi}(n) = \max_{d \in D, |d|=n} \{m \mid \mathcal{A} \text{ solves } d \text{ using } m \text{ work cells } \}$$

is the most amount of space the algorithm $\mathcal{A}$ uses to solve an instance of size n of a problem Π.

We say that $\mathcal{A}$ is a *polynomial-space* algorithm if there exists a polynomial p such that, for all $n \in \mathbb{Z}^+, \mathcal{S}_{\mathcal{A},\Pi}(n) \ll p(n)$. Equivalently, we can say that $\mathcal{A}$ is an *exponential-space* or *logarithmic-space* algorithm.

Similarly to the time property previously defined, we can define space-classes of problems:

Definition 2.6 (PSPACE)**.** *The class PSPACE is the class of problems for which there exists a polynomial-space algorithm which solves it.*

Definition 2.7 (EXPSPACE)**.** *The class EXPSPACE is the class of problems for which there exists an exponential-space algorithm which solves it.*

Definition 2.8 (LOGSPACE)**.** *The class LOGSPACE, or L for short, is the class of problems for which there exists a logarithmic-space algorithm which solves it.*

Note that regarding a logarithmic-space algorithm, only the space used on the work tape is considered.

Nondeterminism

Before providing a detailed description of a Nondeterministic Turing Machine (NDTM), it may be useful to present informally the concept of nondeterministic algorithm. For this purpose, let us define the following problem.

Problem (Traveling Salesman Problem)**.**

- Input: *A graph $G = \langle V, E \rangle$, with V a finite set of vertices and E a finite set of undirected edges, a vertex $s \in V$.*
- Output: *Does there exist a tour starting at s to visit all vertices exactly once ?*

Intuitively, it seems quite hard to create a tour without having to come back on some previous choices as we might only later realize an early mistake. In fact, this problem does not admit a polynomial-time algorithm (unless P = NP). However, if we are given a tour, we can in polynomial-time *verify* whether this tour is valid. Thus, if we were lucky enough at *guessing* a tour, we might find a solution easily. This is the concept at the core of a nondeterministic algorithm.

More formally, a NDTM is structured as an DTM, except for the addition of a *guessing module* attached to a *write-only head*. This module starts on the cell numbered -1 of the work tape. From there, it can either write a symbol in the current cell, move to the left or stop. The choice of steps done by the guessing module are totally arbitrary. When the guessing module becomes inactive, the control starts its computation as in a DTM.

Now, let us assume that given a problem Π, we have a NDTM which, for all instances $d \in D$ in the domain, if $d \in Y$ then there exists a guess that leads the control to answers "yes", and if $d \notin Y$ then there is no such guess. Then we say that this NDTM is a *nondeterministic algorithm* that *solves* the problem Π.

We can now describe the nondeterministic equivalent of the previously introduced complexity classes.

Definition 2.9 (NPTIME)**.** *The class NPTIME, or NP for short, is the class of problems for which there exists a nondeterministic polynomial-time algorithm which solves it.*

Definition 2.10 (NEXPTIME)**.** *The class NEXPTIME is the class of problems for which there exists a nondeterministic exponential-time algorithm which solves it.*

Definition 2.11 (NLOGSPACE)**.** *The class NLOGSPACE, or NL for short, is the class of problems for which there exists a nondeterministic logarithmic-space algorithm which solves it.*

A generalized notation of complexity classes describes the type of Turing machine (deterministic (D), nondeterministic (N)) used along with the bound over the time or space function. For instance, we can equivalently denote NEXPTIME as NTIME($exp(n)$), or LOGSPACE as DSPACE($log(n)$).

An important theorem of the complexity theory is the following:

Theorem 2.1 (Savitch's Theorem [134])**.** *For all function $f(n) \geq log(n)$, NSPACE($f(n)$) $\subseteq$ DSPACE($f^2(n)$).*

In other words, all problems that can be solved by a NDTM in space $f(n)$, can also be solved by a DTM in space $f^2(n)$.

Corollary. *PSPACE = NPSPACE and EXPSPACE = NEXPSPACE.*

Many more complexity classes have been introduced and studied in the literature. However, the problem studied throughout this dissertation will fit inside the presently introduced classes. In Figure 2.2, we depicted the inclusion of the complexity classes. Note that, if a problem can be solved in, let say, polynomial-time then it can be solved by an algorithm in any more *powerful* complexity class, for instance a nondeterministic exponential-time algorithm.

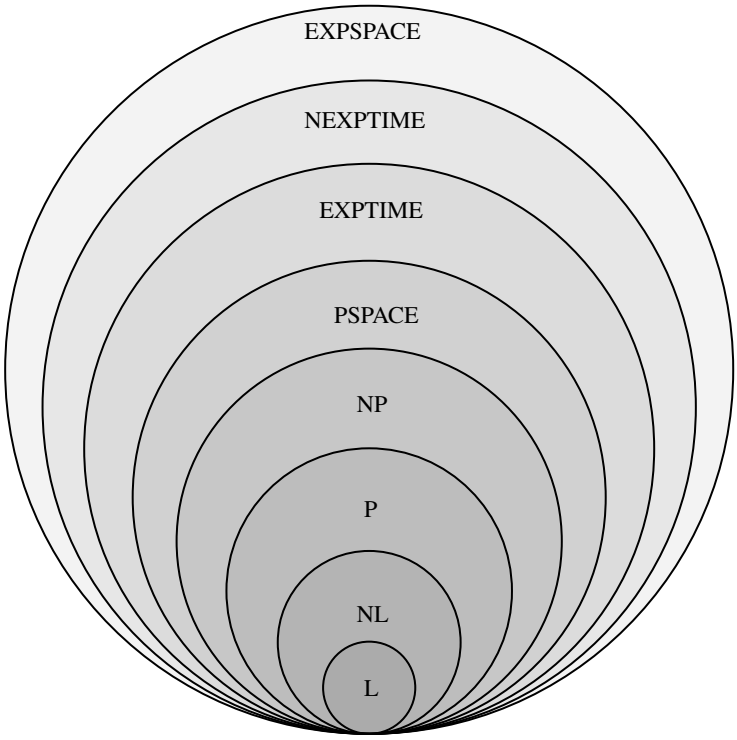


Figure 2.2 – Overview of the complexity classes.

2.1.3 Completeness

Up until now, we described the complexity classes and the membership relation of a problem. A crucial notion for the study of problems is one of completeness. In particular, it gives a tool for studying how "hard" a problem is compared to others. Additionally, if a problem, complete for a specific class, happen to admit an efficient algorithm then any problem of this class can use this algorithm as a subroutine.

The key idea used to show that a problem is complete for a specific class is that of a *transformation*, also called *reduction*. Note that for the sake of conciseness, we will only present *polynomial transformation* that apply to proof of NP-completeness and above. However, an interested reader might want to refer to Sipser [143] and Arora and Barak [7] for details on transformation of P-completeness and below.

Definition 2.12 (Polynomial Transformation)**.** *A polynomial transformation from a problem* Π *to* Π' *is a function* $f : D \mapsto D'$ *that satisfies the following two conditions:*

1. *There exists a DTM that computes f in PTIME;*
2. *For all instances* $d \in D$, $d \in Y$ *if and only if* $f(d) \in Y'$.

Whenever a problem Π has a polynomial transformation to a Π', we say that Π can be polynomially reduced to Π'. Intuitively, it also means that Π' is "at least as hard" as Π. We can now define the concept of completeness.

Definition 2.13 (Completeness)**.** *Given a class $\mathcal{C}$, a problem Π is $\mathcal{C}$-complete if $\Pi \in \mathcal{C}$ and, for all other problem $\Pi' \in \mathcal{C}$, Π can be polynomially reduced to Π'.*

Intuitively, the $\mathcal{C}$-complete problems are the hardest problems in $\mathcal{C}$. From our previous definitions, we can observe that given two problems $\Pi, \Pi' \in \mathcal{C}$, if Π is $\mathcal{C}$-complete and Π reduces to Π' then Π' is $\mathcal{C}$-complete. Thus, throughout this work we will rely on this observation to show that a problem is $\mathcal{C}$-complete. Indeed, we will introduce a problem previously proven $\mathcal{C}$-complete, show that our problem is in $\mathcal{C}$ and show that we have a polynomial transformation from the introduced problem to ours.

Note that the use of a polynomial reduction is not valid to show the completeness of a problem to the class P and below. Indeed, as a polynomial reduction would, by default, require a polynomial amount of time, one must use a logarithmic-space reduction or lower.

2.1.4 Asymptotic notation

We introduced complexity classes which let us prove how hard a problem is. However, when analyzing algorithms it may be important to obtain a more fine-grained classification of the computation time or space. Let us introduce the notion of asymptotic notation for computation time. Nevertheless, the following can be applied to describe spacial behavior of algorithms.

As stated previously, in order to solve a problem, an algorithm will go through a number of steps which may change depending on the instance. Intuitively, consider the problem of sorting a list, in general the algorithm will have to execute more steps to sort a longer list. It is important when analyzing an algorithm to describe how much the number of steps taken grows with the size of the instance.

Let us define multiple notations used to describe the behavior of an algorithm $\mathcal{A}$ on a problem Π.

Definition 2.14 (Big-O Notation)**.** *$\mathcal{A}$ grows in $O(f(n))$ if there exists a positive constant c such that for all n, the number of steps of $\mathcal{A}$ for all instances $d \in D$ of size n is at most $c \times f(n)$.*

Definition 2.15 (Big-Omega Notation)**.** *$\mathcal{A}$ grows in $\Omega(f(n))$ if there exists a positive constant c such that for all n, the number of steps of $\mathcal{A}$ for all instances $d \in D$ of size n is at least $c \times f(n)$.*

Note that $f \in \Omega(g)$ if and only if $g \in O(f)$.

2.2 Multi-Agent Path Finding

Let us present the problem at the origin of this work. In this section, we formally introduce a general definition of the MAPF problem. We present the theoretical setting of the problem and a bounded variant.

2.2.1 Problem Statement

The MAPF is the problem of moving a group of agents in an environment. For theoretical convenience, the environment is modeled as a graph.

Definition 2.16 (Graph)**.** *A graph is a couple $G = \langle V, E \rangle$, with V a set of vertices and $E \in V \times V$ a set of edges.*

Each agent starts at a specific location of the graph and move along the edges to reach their respective targets. In other words, the agents start at a configuration and must reach a provided goal configuration.

Definition 2.17 (Configuration)**.** *A* configuration c *of n agents in a graph G is a vector of elements of V of size n, denoted $c = \langle c_1, \ldots, c_n \rangle$.*

During their travel in the graph, all the agents synchronize when arriving at a node. This sequence of steps is called an execution.

Definition 2.18 (Execution)**.** *An* execution e *of length ℓ with n agents in a graph G is a sequence of configuration $e = \langle c^1, \ldots, c^\ell \rangle$ such that $(c^j, c^{j+1}) \in E$ for all $1 \leq i < \ell$.*

For simplicity, we can note $c^j \rightarrow_G c^{j+1}$ as a shortcut to $(c^j, c^{j+1}) \in E$.

Two types of collisions must be avoided in an execution. Recall that c_k^t denotes the position of agent k at time t.

Definition 2.19 (Vertex Collision)**.** *Two agents a, b are in vertex collision at a time-step t of an execution $e = \langle c^1, \ldots, c^\ell \rangle$ when $c_a^t = c_b^t$.*

Definition 2.20 (Edge Collision)**.** *Two agents a, b are in edge collision at a time-step t of an execution $e = \langle c^1, \ldots, c^\ell \rangle$ when $c_a^{t-1} = c_b^t$ and $c_b^{t-1} = c_a^t$.*

An execution without vertex-collision and edge-collision is said to be *collision-free*. Formally, MAPF can be defined as follows.

Problem 2.1 (MAPF)**.**

- Input: *A graph G and two configurations c^s, c^g of same size.*
- Output: *Does there exist a collision-free execution $e = \langle c^s, \ldots, c^g \rangle$?*

MAPF is a planning problem, thus it is often assumed that a solution always exists. However, it is important to minimize the number of steps in the execution. While multiple definitions of the minimization problem exist, we will restrict our definition to the minimization of the *sum of paths length*. In other words, we consider that an agent increases the cost of the execution if and only if it moves between two configurations or idles on a vertex different from its goal. A bounded version of MAPF can be defined as follows.

Problem 2.2 (Bounded MAPF)**.**

- Input: *A graph G, two configurations c^s, c^g of size n and a bound k.*
- Output: *Does there exist a collision-free execution $e = \langle c^s, \ldots, c^g \rangle$ of cost k?*

CHAPTER 3

STATE OF THE ART

This chapter is dedicated to a summary of the literature surrounding the study of CMAPF and related works. First, we will present MAPF, the complexity of deciding the existence of an unbounded and bounded solution, special cases, variants and algorithms. Then, we introduce the previous contributions on CMAPF and the known results. In addition, we provide a presentation of the studies of the exploration of unknown environments. Finally, we present works which go beyond the initial problem and the possible extensions that our work can undergo.

3.1 Multi-Agent Path Finding

MAPF is a problem which has obtained growing interest in the recent years. Multiple lines of work brought MAPF to its current state. First, Reif [129], in parallel with Hopcroft et al. [71], studied the complexity of the Coordinated Motion Problem which asks for the motion of multiple three-dimensional objects in an environment with obstacles. This problem was shown to be PSPACE-hard, even in the two-dimensional case. An example of such a problem is the case of two disc-shaped robots moving in the presence of polygonal obstacles [136]. The planning problem for multiple agents is PSPACE-hard even when robot shapes are restricted to simple tiles [67]. PSPACE-hardness was shown in case of unit-square robots with polygonal obstacles as well when agents are *unlabelled/homogeneous*, or in our terminology, anonymous [144]. A line of algorithms that avoid this high complexity are those based on sampling methods such as Kavraki et al. [79] and Kuffner and LaValle [83]. They randomly sample points and check whether these points can be connected respecting the constraints of the system. This allows one to quickly generate a tree or a graph between sampled points, and continue sampling points until the desired plan is found.

In another line of work, Ratner and Warmuth [128] studied the generalized version of the 15-Puzzle. They showed that finding the shortest solution to this problem is NP-hard. In order to deal with the high complexity of the problem, multiple approaches were investigated.

First, with the introduction of A* [65], the *centralized* approach was investigated [86]. In this algorithmic approach, the group of agent is considered to be a single entity. Many variants of A* were studied and introduced to solve the MAPF efficiently and/or optimally. A standard of the game industry was Local Repair A* (LRA*) [148]. However, many more improvements and variants were studied [145, 58].

Another approach called the *decoupled* approach, recently gave birth to the most efficient algorithm to solve MAPF. The strategy of such approach is to consider each agent as independent, and plan their paths separately. A good example of such a technique is Windowed Hierarchical Cooperative A* (WHCA*) [141]. Agents are considered in turns; each one is assigned a path

avoiding collisions with previously assigned agents. Similar approaches were studied by Dresner and Stone [39], Wang, Botea, et al. [159], and Jansen and Sturtevant [77]. However, contrarily to the centralized approaches, these are not optimal. In order to solve this last issue, Sharon et al. [139] introduced CBS which was improved with many powerful optimizations [138, 19, 46, 91, 90].

3.1.1 Complexity

The complexity of MAPF has been intensively studied in order to find practical classes in which the problem is easier or that can be used to simplify the computation of the general problems through heuristics. Indeed, MAPF was studied on planar graphs [167], on grid graphs [171]. While no practical classes of graphs seem to yield a tractable variant of the decision problem associated to the optimization of MAPF, deciding the existence of an execution was shown to be in PTIME [169].

Slideable

An interesting restriction of MAPF on grid maps is the one of *slideable* instances. This class of instances, introduced by Wang and Botea [160], admits a polynomial-time algorithm. For an instance to be Slideable, there must exist a path for all agents to their goals which at all time admits alternative moves. The core idea is that the algorithm does not require replanning. While this algorithm is not complete for general MAPF, it runs in $O(|V|^2 n^2)$ or even linearly on some instances.

3.1.2 Variants

Here we present multiple variants of MAPF that have been studied to capture additional important concepts.

Target Assignment

Introduced by Ma and Koenig [102], the Target-Assignment and Path Finding (TAPF) problem is the problem of assigning targets to the agents in teams along planning collision-free paths. This problem generalizes both the *anonymous* and *non-anonymous* MAPF. Indeed, the anonymous MAPF would be TAPF with a single team of agents, and the non-anonymous MAPF would be TAPF with single agent teams. Multiple algorithmic solutions were introduced to solve TAPF [102, 114, 99].

Robustness

The theoretical formulation of MAPF does not take into account the various mishaps that can happen during the execution of the plan in the real world. Multiple concepts of robustness have been studied to create plans which can react or avoid problems during the execution. The

concept of k-robustness [11], allows the agents to *survive* to a bounded number of mishaps. A parallel line of work studied a probabilistic robustness [10], which can provide a solution which succeeds with probability at least p.

Lifelong

While MAPF solves the problem of efficiently moving a group of agents in an arbitrary complex environment, it does not handle an important characteristic of the *warehouse problems*. In such a setting, the system continuously receives new requests of targets for the agents to reach. The Lifelong MAPF, introduced as Multi-Agent Pickup and Delivery (MAPD) [104], solves this *online* version of MAPF. More works on this particular variant was done by Ma et al. [100] and Li et al. [94].

3.2 Coverage Planning

The coverage path planning problem has been studied in different settings; see the surveys of Choset [32], Galceran and Carreras [54], and Cabreira et al. [25]. This problem can be used to solve inspection (where a given set of points of interests must be observed), surveillance (points of interest must be continuously visited), and can be used in applications such as lawn mowing and floor cleaning. One can prove probabilistic completeness results in some cases, which means that if a solution exists, then the algorithm will eventually find it with probability 1; although it may not be able to detect that no solution exists [42]. Some of these algorithms work in two phases: the first phase consists in sampling points and checking the feasibility of the edges between them, and the second phase consists in solving a graph problem on this structure, and repeating the first phase if no solution is found. Some works use approximations of metric Traveling Salesman Problem (TSP) to find solutions in the constructed graph [35, 41]. Other works use A^* search algorithms with suitable heuristics to compute approximate solutions [53].

Several works consider the coverage path planning problem for multiple agents [31, 54, 25]. Kusnur et al. [85] introduced a centralized framework to compute plans for the persistent coverage problem without communication constraints. In Rekleitis et al. [131] gave algorithms for the repeated coverage problem in unknown environments for a team of robots, including the case of the line-of-sight communication restriction. This uses the cell decomposition given by Choset and Pignon [33] for the coverage path planning. Hazon and Kaminka [66] and Xiaoming Zheng et al. [163] obtain a graph structure thanks to a cell decomposition, and variants of spanning trees are computed on this graph to ensure coverage with multiple robots. Graph problems related to this problem are studied by Xu [164]. The generalization of TSP was studied by Anbuudayasankar et al. [5] which is a relevant problem for coverage path planning in the multi-agent setting. Some works also consider maneuverability and camera angle constraints [1].

3.3 Connectivity

The continuous communication restriction appears in several works which provide experiments to demonstrate the feasibility of the proposed algorithms [120, 113, 166, 152]. Some works consider the use of dynamic teams of robots which exchange their information on current solutions [118]. Communication and battery restrictions are considered together by Cesare et al. [26] which gave an algorithm in which robots can adopt the roles for exploring, meeting, sacrificing themselves (continuing the mission in case of low battery), and serving as communication relay for other robots. Different communication restrictions have been considered as well such as event-based communication where the discovery of new information triggers communication (see *e.g* [3, 13]).

3.3.1 Connected MAPF

The CMAPF problem was initially introduced by Hollinger and Singh [69] along with an online algorithm. The algorithm would run on each agent separately and guarantee a periodic reconnection of the agents. The NP-hardness of bounded CMAPF was also proven in this work. Later, Tateo et. al. [151] provided a complexity analysis of the unbounded problem and different variants. The problem is shown PSPACE-complete without a bound, regardless of the collision-free requirement, or the addition of a base station. Here, we remind the reader the previous results of CMAPF.

Theorem 3.1 ([69])**.** *Deciding the existence of a bounded connected execution on undirected topological graphs is NP-hard.*

Theorem 3.2 ([151])**.** *Deciding the existence of a connected execution on undirected topological graphs is PSPACE-complete.*

3.4 Partially-Known Environment

The exploration and navigation in a partially-known/unknown environment has been intensively studied. Many parallel lines of works studied different approaches and model of this problem. Let us present a few from which our framework is inspired.

3.4.1 SLAM

The Simultaneous Localization and Mapping (SLAM) problem asks to integrate the information collected during navigation into the most accurate map possible. A good overview of the many methods developed for unknown environment mapping can be found in the work of Rao et al. [126]. However, SLAM does not address the navigation task. Two different problems emerged to address the planning of the navigation in parallel to the mapping task. The Simultaneous Planning Localization and Mapping (SPLAM) asks for a motion plan through the unknown environment, which is quite intractable [84, 88, 22]. On the other hand, the Next-Best

View (NBV) problem [60, 116, 4] asks to guide a robot through the environment to optimize the amount of new information gathered per step.

3.4.2 Canadian Traveler Problem

An interesting variant of the Shortest Path problem is the Canadian Traveler Problem (CTP) initially formulated by Papadimitriou and Yannakakis [121]. A traveler wishes to travel to a city during winter. However, it might happen that some roads have not been cleared of snow. Thus, it is assumed that when arriving at an intersection, the traveler is able to observe which roads are clear. The problem has recently received additional interest [15, 115, 165, 44].

PART I

CONNECTIVITY

CHAPTER 4

CONNECTED MULTI-AGENT PATH PLANNING

4.1 Introduction

The MAPF problem asks for a plan to move a group of agents to a target configuration in a graph while avoiding collisions. It is an important problem in the design of groups of autonomous vehicles, and has been used in several applications such as Kiva (Amazon Robotics) warehouse systems [162], autonomous aircraft towing vehicles [110], office robots [157] and characters in video games [142].

A closely related problem is that of coverage path planning which consists in computing a plan that visits a set of given locations in a graph. A comprehensive survey is provided by Galceran and Carreras [54]. Applications include underwater ship hull inspection [42], wildfire tracking with drones [123], to name a few.

An important application area is that of information gathering missions in which agents must visit a set of locations in an area and gather information using sensors (e.g. camera, smoke sensor, hygrometer etc.). Some applications, such as search and rescue missions, might require continuous connection between all agents and a base station, for instance, in order to stream video and allow human operators to make decisions [3]. In this case, the path planning and coverage path planning algorithms must take additional connectivity constraints into account in order to compute suitable plans. Several works have explored planning algorithms in this setting *e.g.* [132, 120].

The variant of MAPF with connectivity constraints and related computational complexity results were studied by Hollinger and Singh [69] and Tateo et al. [151]. The latter present complexity results for connectivity constraints with and without collision constraints and that the existence of a plan of arbitrary length in both cases is PSPACE-complete in undirected graphs. Interestingly, it means that it has the same complexity as classical planning [24]. The work of Bodin et al. [18] considers the coverage path planning and provides experiments in which instances are described in Planning Domain Definition Language (PDDL) and solved with the planner Functional STRIPS [52]. In other words, MAPF with connectivity constraints is more difficult than MAPF with collision constraints: deciding the existence of a collision-free plan of arbitrary length is in PTIME, as stated by Yu and Rus [172].

Now, concerning the optimization problems, there is a subtlety about the encoding (unary vs binary) of the length of plans. For MAPF with collision constraints, as the existence of a plan is equivalent to the existence of a plan of length $O(|V|^3)$ where $|V|$ is the number of nodes [172], the encoding of the length is not relevant. Thus, several papers on the topic do not specify the

encoding [14, 105, 128, 167, 168, 169]. However, for MAPF with connectivity constraints, there is no such bound on the length of a plan. We show that the existence of a bounded plan when the bound is given in binary is PSPACE-complete; while when the bound is given in unary, the problem is NP-complete, even on undirected graphs, as claimed by Hollinger and Singh [69], although they do not explicitly specify the membership proof and are ambiguous about the encoding [1].

We mainly consider a setting where collisions are ignored. In fact, we are interested in the computational complexity of maintaining connectivity in plans. As done by Hollinger and Singh [69], agents are be assumed to be equipped with a low-level collision avoidance system in some cases. Additionally, in drone applications with few agents, different altitudes can be used to avoid collisions. Nevertheless, we do discuss the impact of taking collision constraints into account in our results as well.

In this chapter, we are interested in the computational complexity of two problems: **CMAPF** is the variant of MAPF with connectivity constraints, without collision constraints; and **CMACP** is the variant of coverage path planning for multiple agents with connectivity constraints. As in the work of Hollinger and Singh [69] and Tateo et al. [151], a problem instance is a *topological graph*, which is a set of nodes given with *movement edges* along which agents can move, and *communication edges* which determine pairs of vertices at which communication is possible. Our results are as follows. We establish the computational complexity of determining the existence of plans, showing that **CMAPF** is PSPACE-complete on directed graphs as well, and proving that **CMACP** is PSPACE-complete on both types of graphs. We study bounded versions of these problems where a bound on the length of the plan is given as part of the input; these are denoted by **bCMAPF** and **bCMACP**. As done by Turner [155], we advocate for lengths of plans written in unary, although we also study the complexity of our problems when the encoding is binary. We show that both **bCMAPF** and **bCMACP** are NP-complete when the bound is given in unary, and we clearly state their PSPACE-completeness when the bound is given in binary.

Given the prohibitively high complexity reported above, we are interested in searching for 'easier' classes of topological graphs which are realistic for the purpose of information gathering missions. One of our main contributions is the identification of a natural class of topological graphs, called *sight-moveable*, for which we give efficient algorithms. This class requires that whenever an agent can communicate with another node, then it can also move to that node while maintaining direct communication. This can be seen as a restriction on allowed topological graphs; however, if the graph at hand does not have this property, it may be possible to enforce it by removing some communication edges (not allowing the planning algorithm to rely on those edges). Subsection 4.5.3 describes a way to obtain sight-moveable graphs from a given topological graph. Thus, any plan found in the obtained sight-moveable graph can be applied on the original one.

The class of sight-moveable graphs offers good computational properties: both **CMAPF** and **CMACP** belong to NLOGSPACE, meaning that they can be solved by a non-deterministic

1. The bounded plan is clearly in NP when the bound ℓ is written in unary: simply guess a solution of length at most ℓ and check that it is correct. When the bound is given in binary, a solution may be in exponential size, and cannot be guessed in polynomial time.

algorithm that only uses a logarithmic amount of memory. Practically, it means that algorithms for generating plans can be parallelized, because NLOGSPACE is included in the Nick's class (NC), [34], known to represent decision problems for which there is a parallel algorithm. However, the bounded versions remain NP-complete. We complete the investigation with the complexity analysis of the problem with complete communication graphs, where all pairs of nodes can communicate. In addition, we give complexity results on several variants of our problems.

Note that the PSPACE lower bound given by Tateo et al. [151] concerned the reachability problem where agents start from arbitrary locations, whereas we prove that the PSPACE lower bound still holds for agents starting all from the base. Furthermore, this chapter extends the work of Charrier et al. [29] and Charrier et al. [30], in which the complexity of **CMAPF** (when agents start from the base) and **CMACP** on undirected topological graphs was left open. In this contribution, we solved them (Theorem 4.3 and 4.4). We also include detailed proofs and discuss relevant extensions that were not considered in previous work.

Overview In Section 4.2, we introduce the required notions for the rest of the chapter. In Section 4.3, we describe the upper bounds of our problems on directed topological graphs and, in Section 4.4, we prove the lower bounds on undirected topological graphs to obtain completeness results. In Section 4.5, we study sight-moveable topological graphs. Section 4.6 contains the complexity analysis of the complete-communication topological graphs. We introduce relevant extensions of our problems, in Section 4.7. We present a conclusion and future works in Section 4.8.

4.2 Preliminaries

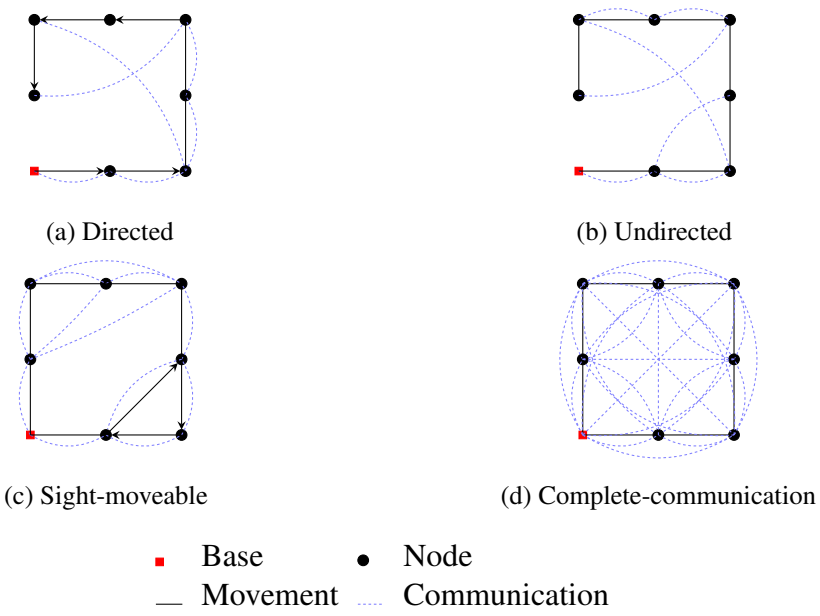
In some path finding applications, one considers a discretization of the space which yields a graph of movements on which algorithms are run. For instance, *regular grids* which decompose the space in square, triangular or hexagonal cells, *irregular grids* with techniques such as *quadtrees* [51, 81] or *Voronoï diagrams* comprehensively discussed in the survey of Aurenhammer [12].

Our work is independent of the particular method used to obtain the discretization. We only work under the hypothesis that a feasible plan on the graph generated by the discretization is also feasible in the continuous space.

We will, now, introduce the notions used throughout this work. We define formally the general topological graphs and some subclasses, the notion of execution and the properties considered, the decision problems we investigate and some known results on these problems.

4.2.1 Topological Graphs

Our problems require graphs with two types of edges: *movement edges* along which agents can move, and *communication edges* which specify whether agents at two different locations can communicate. We call graphs with this additional information *topological graphs*. The formal definition is the following and examples are depicted in Figure 4.1.



(a) Directed (b) Undirected

(c) Sight-moveable (d) Complete-communication

Figure 4.1 – Examples of topological graphs.

Definition 4.1 (Topological Graph)**.** *A* topological graph *is a tuple* $G = \langle V, E_m, E_c \rangle$*, with* V *a finite set of nodes containing a distinguished node* B *called the base,* $E_m \subseteq V \times V$ *a set of movement edges and* $E_c \subseteq V \times V$ *a set of undirected communication edges.*

The node B is the supervision base station from which the agents start the mission and with which they are required to keep communication.

In some applications movement edges are reversible, that is, if an agent can travel from a node to another, it can also go back to the former through the same edge. Undirected graphs thus naturally arise in some applications. See Figure 4.1b for an example.

Definition 4.2 (Undirected Topological Graph)**.** *A topological graph is said to be* undirected *if* $\langle V, E_m \rangle$ *is an undirected graph.*

Let us now introduce our new class, called *sight-moveable* topological graphs, which is one of our main contributions. This class requires the movement edges to be reflexive. In addition, whenever an agent can communicate with another node, then it can also move to that node while maintaining the communication with the node it started from. An example of a sight-moveable graph is given in Figure 4.1c.

Definition 4.3 (Sight-Moveable Topological Graph)**.** *A* sight-moveable *topological graph* $G = \langle V, E_m, E_c \rangle$ *is a directed topological graph such that*

1. $E_m \subseteq E_c$*,*
2. *for all* $v \in V$*,* $(v, v) \in E_m$*,*
3. *and whenever* $(v, v') \in E_c$*, there exists a sequence* $\rho = \langle \rho_1, \ldots, \rho_n \rangle$ *of nodes such that* $v = \rho_1$*,* $v' = \rho_n$*, for all* $i \in \{1, \ldots, n-1\}$ $(v, \rho_i) \in E_c$ *and* $(\rho_i, \rho_{i+1}) \in E_m$*.*

Last, we define the *complete-communication* topological graphs which are simply sight-moveable topological graphs with undirected movement and complete communication topology. An example of such a graph is depicted in Figure 4.1d, and the formal definition is the following.

Definition 4.4 (Complete-Communication Topological Graph)**.** *A topological graph is said to be a* complete-communication *if it is a sight-moveable topological graph such that $\langle V, E_m\rangle$ is an undirected graph and $E_c = V \times V$.*

Observe that a complete-communication graphs are reflexive, undirected, connected graphs with communication edges between each pair of nodes.

We say that a topological graph has a *planar (grid) movement graph* iff the graph $\langle V, E_m\rangle$ is a planar (resp. grid) graph.

4.2.2 Execution

An *execution* is a finite sequence of *configurations* describing the positions of the agents during the mission. We require that all agents should be connected to the base in all configurations. We will use multi-sets to denote nodes occupied by agents in configurations, since agents are *anonymous*. In other terms, if the goal is to reach a target configuration, it does not matter which agent occupies which node, as long as there is the right number of agents at each node. Other works use the term *unlabelled* or *homogeneous* (see *e.g.* [144]); but we use the terminology of Multi-Agent Path Finding (MAPF).

The formal definition of a configuration is the following.

Definition 4.5 (Configuration)**.** *A* configuration *c of n agents in a topological graph G is a tuples of elements of V of size n, denoted $c = \langle c_1, \ldots, c_n\rangle$.*

A configuration is said to be *connected* iff the graph $\langle V_c, E_c \cap (V_c \times V_c)\rangle$ is connected with $V_c = \{B, c_1, \ldots, c_n\}$.

Given a topological graph $G = \langle V, E_m, E_c\rangle$, we write $c \rightarrow_G c'$ to say that agents in c perform one-step movements to occupy nodes in c'. Formally, we have $c \rightarrow_G c'$ if c (resp. c') can be written as $\langle c_1, \ldots, c_n\rangle$ (resp. $\langle c'_1, \ldots, c'_n\rangle$), and $(c_i, c'_i) \in E_m$ for all $1 \leq i \leq n$.

Definition 4.6 (Execution)**.** *An* execution *e of length ℓ with n agents in a topological graph G is a sequence of connected configuration $e = \langle c^1, \ldots, c^\ell\rangle$ such that $c^j \rightarrow_G c^{j+1}$ for all $1 \leq j < \ell$.*

In our setting, the *makespan* of an execution is equal to its length.

A *covering execution* $e = \langle c^1, \ldots, c^\ell\rangle$ of length ℓ with n agents in a graph G is an execution such that $c^1 = c^\ell = \langle B, \ldots, B\rangle$ and for all $v \in V$, there exists $j \in \{1, \ldots, \ell\}$ such that v appears in c^j. An example of such an execution is depicted in Figure 4.2 (from left to right).

4.2.3 Decision Problems

We formally define the problems **CMAPF** and **CMACP** and their bounded versions **bCMAPF** and **bCMACP**.

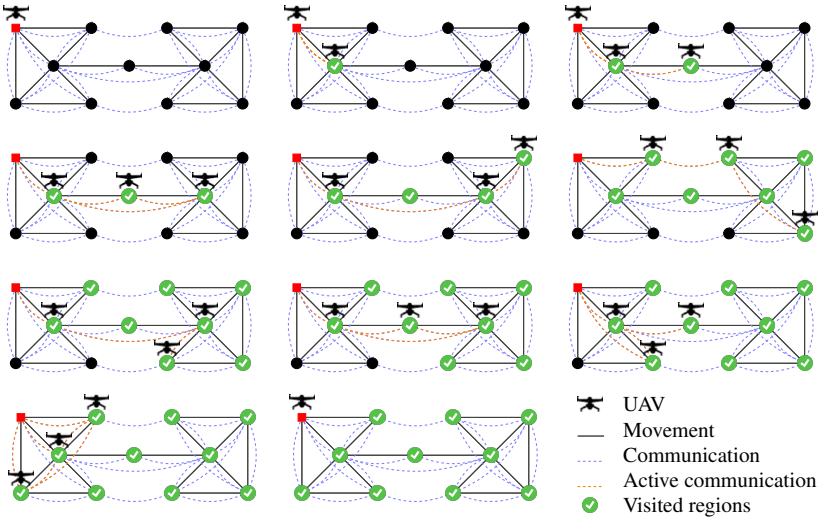


Figure 4.2 – Example of a covering execution of length 10 with 3 agents. The plan is depicted from left to right and from top to bottom.

The Unbounded Case

Definition 4.7 (CMAPF). *Given a topological graph G, an integer n and a configuration c of size n, decide if there is an execution $\langle c^1, \dots, c^\ell \rangle$ in G such that $c^1 = \langle B, \dots, B \rangle$ and $c^\ell = c$.*

Definition 4.8 (CMACP). *Given a topological graph G and an integer n, decide if there exists a covering execution with n agents.*

We also consider variants **CMAPF-init** and **CMACP-init** denote the problems in which the agents start at a given configuration rather than at the base. In other words, the initial configuration is part of the input. In addition, **CMACP-init** requires the agents to return to the initial configuration rather than to the base.

In the above problems, the encoding of the integer n (unary or binary) does not matter. Indeed, in **CMAPF**, **CMAPF-init** and **CMACP-init**, the input already contains some configuration which is of size n; in **CMACP**, it is useless to have n greater than the number of nodes.

The Bounded Case

The bounded versions are inspired from the so-called polynomial-length planning problem [155] in which we ask for the existence of a plan of length bounded by a polynomial in the size of the planning task. This can be seen as the decision problem for the optimization problem that seeks to minimize the length of an execution, except that we assume that the bound is given in *unary*. In fact, the goal of planning algorithms is to compute plans, so given a bound ℓ on the length of the desired plan, the algorithm always allocates memory space of size $\Omega(\ell)$ to store the plan.

That is why we use unary encoding in the following definitions. Binary encoding of the length ℓ is discussed as well in Subsection 4.7.1.

Definition 4.9 (**bCMAPF**). *Given a topological graph G, an integer n and a configuration c of size n and ℓ an integer written in unary, decide if there is an execution $\langle c^1, \ldots, c^{\ell'} \rangle$ in G s.t. $\ell' \leq \ell$, $c^1 = \langle B, \ldots, B \rangle$ and $c^{\ell'} = c$.*

Definition 4.10 (**bCMACP**). *Given a topological graph G, an integer n, an integer ℓ written in unary, decide if there exists a covering execution of length ℓ' such $\ell' \leq \ell$.*

Restriction to Subclasses of Graphs

We consider the restriction of the above problems to the following subclasses of graphs: directed graphs (denoted by DIR), undirected graphs (denoted by UND), sight-moveable graphs (denoted by SM) and complete-communication graphs (denoted by CC). The variants of these problems to a given graph class will be denoted using a subscript, that is, **CMAPF**$_\star$, **CMACP**$_\star$, **bCMAPF**$_\star$, **bCMACP**$_\star$, **CMAPF-init**$_\star$, and **CMACP-init**$_\star$ denote the restriction of these problems to graphs of type $\star \in \{\text{DIR}, \text{UND}, \text{SM}, \text{CC}\}$.

4.2.4 Known Results

The connected version of MAPF was introduced by Hollinger and Singh [69], in which a topological graph discretizes the space and it is proved that the existence of a plan for the reachability of a configuration of non-anonymous agents in a bounded number of steps with collisions allowed is NP-hard:

Theorem 4.1 ([69]). **bCMAPF-init**$_{\text{UND}}$ *is NP-hard.*

As stated before, the above paper actually states the NP-hardness of this problem but without specifying the encoding of the bound.

Tateo et al. [151] establish the complexity of **CMAPF-init**$_{\text{UND}}$:

Theorem 4.2 ([151]). **CMAPF-init**$_{\text{UND}}$ *is PSPACE-complete.*

We relate both problems showing PSPACE-hardness when agents all start at the base.

4.2.5 Overview of Results

In the rest of the work, we study upper and lower complexity bounds for the defined decision problems on different topological graphs. The following sections present our results, respectively, for the general case, the undirected graphs, sight-moveable graphs, and complete-communication graphs. An overview of these results is given in Figure 4.1.

<table>
<tr><th>Class/Problem</th><th>**CMAPF**
(Def. 4.7)</th><th>**CMACP**
(Def. 4.8)</th><th>**bCMAPF**
(Def. 4.9)</th><th>**bCMACP**
(Def. 4.10)</th></tr>
<tr><td>Directed
(Def. 4.1)</td><td>PSPACE-complete
(Th. 4.3)</td><td rowspan="2">PSPACE-complete
(Th. 4.4)</td><td rowspan="2">NP-complete
[69]</td><td rowspan="4">NP-complete
(Th. 4.8)</td></tr>
<tr><td>Undirected
(Def. 4.2)</td><td>PSPACE-complete
[151]</td></tr>
<tr><td>Sight-Moveable
(Def. 4.3)</td><td rowspan="2">in LOGSPACE
(Prop. 4.3)</td><td rowspan="2">in NLOGSPACE
(Prop. 4.4)</td><td>NP-complete
(Th. 4.6)</td></tr>
<tr><td>Complete-Comm.
(Def. 4.4)</td><td>in NLOGSPACE
(Prop. 4.5)</td></tr>
</table>

Table 4.1 – Overview of the complexity results.

4.3 Directed Topological Graphs

In this section, we will consider the previous problems restricted to the class of directed topological graphs. These problems are thus denoted by $\mathbf{CMAPF}_{\text{DIR}}$ and $\mathbf{bCMAPF}_{\text{DIR}}$ (resp. $\mathbf{CMACP}_{\text{DIR}}$ and $\mathbf{bCMACP}_{\text{DIR}}$), with DIR denoting the class of directed topological graphs.

In this section, we show upper bounds for all our problems in the general case, that is, for directed topological graphs. Observe that this also provides upper bounds for other classes such as undirected graphs.

For the unbounded problems, we can design a straightforward non-deterministic algorithm running in polynomial space, that guesses an execution by keeping in memory the last configuration, and, for **CMACP**, the set of visited regions. In fact, the number of configurations is exponential, and a single configuration can be stored in polynomial space. Moreover, one can easily bound the length of executions by an exponential as well. We conclude with Savitch's Theorem (NPSPACE=PSPACE)[134]:

Proposition 4.1. **CMACP** *and* **CMAPF** *are in PSPACE.*

For the bounded versions of the problems, since the bound is encoded in unary, one can guess and check a path of bounded length in polynomial time. The result follows.

Proposition 4.2. **bCMACP** *and* **bCMAPF** *are in NP.*

4.4 Undirected Topological Graphs

In this section we prove the PSPACE lower bound of the problems **CMAPF** and **CMACP** on undirected topological graphs.

We consider the result of Theorem 4.2 in the setting where all agents start at the base.

Theorem 4.3. $\mathbf{CMAPF}_{\text{UND}}$ *is PSPACE-complete.*

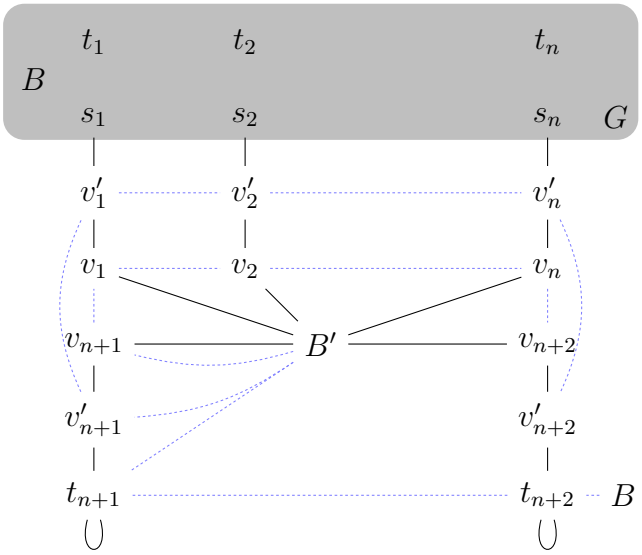


Figure 4.3 – Reduction of **CMAPF-init**$_{\text{UND}}$ into **CMAPF**$_{\text{UND}}$. The node t_{n+2} communicates with all nodes v such that v communicates with B.

Proof. The upper bound comes from Proposition 4.1. The lower bound is by reduction from **CMAPF-init**$_{\text{UND}}$ (see Theorem 4.2). Let us denote by a tuple (G, B, n, s, t) the instances of **CMAPF-init**$_{\text{UND}}$ where G is the graph, B the base, n the number of agents, s the initial configuration and t the target configuration. Instances of **CMAPF**$_{\text{UND}}$ will be denoted by (G, B, n, t) as the initial configuration is fixed.

Let (G, B, n, s, t) be an instance of **CMAPF-init**$_{\text{UND}}$. We show how to map (G, B, n, s, t) to an instance of **CMAPF**$_{\text{UND}}$ in polynomial time. We construct the instance $(G', B', n+2, t')$ of **CMAPF**$_{\text{UND}}$ where G' is given in Figure 4.3, B' is the base, and the final configuration t' is $\langle t_1, \ldots, t_n, t_{n+1}, t_{n+2} \rangle$.

Let us describe more precisely the construction of G'. We write $s = \langle s_1, \ldots, s_n \rangle$ and $t = \langle t_1, \ldots, t_n \rangle$. The graph G' contains the graph G: in particular, it contains B that is no longer the base; a new node B' is now the base. Let us describe the construction of G'.

- *Nodes* We first create two layers of $n+2$ vertices. The first layer is composed of the following vertices $v_1, \ldots, v_n, v_{n+1}, v_{n+2}$ and the second of $v'_1, \ldots, v'_n, v'_{n+1}, v'_{n+2}$. We also add two nodes t_{n+1} and t_{n+2}.
- *Movement* We add movement edges between t_{n+1} and v'_{n+1}, t_{n+2} and v'_{n+2}, t_{n+1} and t_{n+1}, and t_{n+2} and t_{n+2}. Then, the role of t_{n+1} and t_{n+2} is to relay the communication from B' to nodes in G. We connect B' to the first layer, i.e. with a movement edge between B' and v_i, for all $1 \leq i \leq n+2$. The first layer has movement edges to the second layer, i.e. with a movement edge between v_i and v'_i, for all $1 \leq i \leq n+2$. The n first vertices of the second layer has movement edges to the initial configuration s such that there is a movement edge from v'_i to s_i, for all $1 \leq i \leq n$.
- *Communication* We add communication edges from B' to t_{n+1}, from t_{n+1} to t_{n+2} as well as from t_{n+2} to B and if there exists v such that B communicates v then we create

a communication edge from t_{n+2} to v. We add a communication edge from B' to v_{n+1}, from v_i to v_{i+1}, for all $1 \leq i < n$, from v_{n+1} to v_1 and from v_n to v_{n+2}. We repeat this last procedure for the second layer as well.

We now give the formal definition of our reduction. In the sequel, we use the symbol $\sqcup$ to emphasize that the union is of *disjoint* sets. Formally, given (G, B, n, s, t) with $G = \langle V, E_m, E_c \rangle$ with base B, we define $(G', B', n+2, t')$ where $G' = \langle V', E'_m, E'_c \rangle$ with base B' where:

– $V' := V \sqcup \Big\{B', v_1, \ldots, v_n, v'_1, \ldots, v'_n, v_{n+1}, v_{n+2}, v'_{n+1}, v'_{n+2}, t_{n+1}, t_{n+2}\Big\}$

– E'_m is the symmetric closure of
$E_m \cup \{(B', v_1), \ldots, (B', v_n), (B', v_{n+1}), (B', v_{n+2})\}$
$\cup \{(v_1, v'_1), \ldots, (v_n, v'_n)\} \cup \{(v'_1, s_1), \ldots, (v'_n, s_n)\}$
$\cup \Big\{(v_{n+1}, v'_{n+1}), (v_{n+2}, v'_{n+2}), (v'_{n+1}, t_{n+1}), (v'_{n+2}, t_{n+2})\Big\}$
$\cup \{(t_{n+1}, t_{n+1}), (t_{n+2}, t_{n+2})\}$;

– E'_c is the symmetric closure of
$E_c \cup \Big\{(v'_1, v'_2), \ldots, (v'_{n-1}, v'_n)\Big\} \cup \{(v_1, v_2), \ldots, (v_{n-1}, v_n)\}$
$\cup \Big\{(B', v_{n+1}), (B', v'_{n+1}), (B', t_{n+1})\Big\}$
$\cup \Big\{(v'_1, v'_{n+1}), (v'_n, v'_{n+2}), (t_{n+2}, B)\Big\}$
$\cup \{(t_{n+2}, v) \mid (B, v) \in E_c\}$
$\cup \{(v_1, v_{n+1}), (v_n, v_{n+2}), (t_{n+1}, t_{n+2})\}$.

It is worth noting that all connected configurations in G are now connected via B', t_{n+1} and t_{n+2}. We now show the instance (G, B, n, s, t) of **CMAPF-init**$_{\text{UND}}$ is feasible if, and only if in the constructed instance $(G', B', n+2, t')$ of **CMAPF**$_{\text{UND}}$ is feasible.

($\Rightarrow$) Suppose that t is reachable from s in the instance (G, B, n, s, t). We construct an execution for $(G', B', n+2, t')$ as follows. All agents start from B'. The $n+2$ agents first reach $\langle v_1, \ldots, v_{n+2} \rangle$, then $\langle v'_1, \ldots, v'_{n+2} \rangle$, then $\langle s_1, \ldots, s_n, t_{n+1}, t_{n+2} \rangle$. After that, the agents at positions $s_1, \ldots, s_n$ reach positions $t_1, \ldots, t_n$ (following the same plan as in for (G, B, s, t)), while the two others remain in t_{n+1} and t_{n+2}.

($\Leftarrow$) Let us consider an execution e from $\langle B', \ldots, B' \rangle$ to t' for $(G', B', n+2, t')$. Let us extract an execution from s to t for (G, B, n, s, t). In order to do so, we prove the Facts 1 and 2.

Fact 1. *The configuration $\langle s_1, \ldots, s_n, t_{n+1}, t_{n+2} \rangle$, up to a permutation, appears in the execution e.*

Proof. The configuration t' is reached at the end of the execution e and the $n+2$ agents started at B'. Thus, n agents must enter into G and be at one of the s_i at some point. Let us consider the moment of the execution e when an agent, denoted a, occupies a node s_i. For that, it must be at v'_i before going to s_i. Furthermore, for agent a to be connected at v'_i the nodes v'_{n+1} and $v'_1, \ldots, v'_{i-1}$ must be occupied. At the next step, at least the nodes t_{n+1} and t_{n+2} must be occupied for agent a to be connected at s_i.

Consider the last instant before t_{n+2} is first occupied (some instant where t_{n+2} is occupied has to exist since otherwise no visit of G coud be connected to B'). At this last instant some agent is necessarily at v'_{n+2}, hence $v'_1, \ldots, v'_{n+1}$ are also occupied.

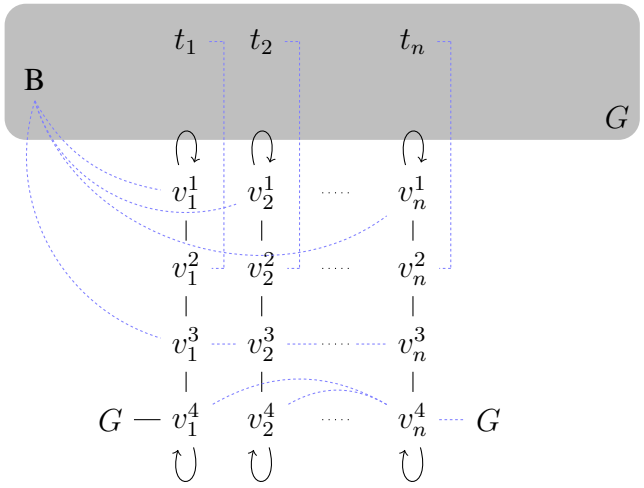


Figure 4.4 – Reduction of **CMAPF**$_{\text{UND}}$ to **CMACP-init**$_{\text{UND}}$.

Towards contradiction, assume the agent at v'_i next moves to v_i. Then for the connection to be maintained from B' to this agent, also (in particular) v_{n+1} has to be occupied; this is necessarily by the (necessarily unique) agent which was at v'_{n+1}, hence t_{n+1} is not occupied and so, the agent at t_{n+2} is not connected, contradiction. Hence the next configuration must be $\langle s_1, \ldots, s_n, t_{n+1}, t_{n+2}\rangle$. △

Now, by Fact 1, we can consider the last time at which the agents are in the configuration $\langle s_1, \ldots, s_n, t_{n+1}, t_{n+2}\rangle$. We prove the Fact 2.

Fact 2. *Between that last time and the end of the execution, there are always the same n agents in G, one agent on t_{n+1} and one on t_{n+2}.*

Proof. A similar reasoning to the last one can be used to show that if an agent is located at s_i before moving to v'_i then nodes from s_1 to s_n as well as t_{n+1} and t_{n+2} are occupied. Thus, since this is the last time the agents are in the configuration $\langle s_1, \ldots, s_n, t_{n+1}, t_{n+2}\rangle$, none of the n first agents can move out of G. Furthermore, t_{n+1} and t_{n+2} must stay occupied in order for the n agents in G to be connected to B'. △

From Facts 1 and 2, the positions of the first n agents in the portion of the execution between the last time in $\langle s_1, \ldots, s_n, t_{n+1}, t_{n+2}\rangle$ and t' are fully in G and gives an execution starting at s and finishing at t for (G, B, n, s, t). □

We now turn our attention to **CMACP**$_{\text{UND}}$. To do so, we start by establishing the PSPACE-completeness of **CMACP-init**$_{\text{UND}}$, and then show how to reduce this problem to **CMACP**$_{\text{UND}}$.

Lemma 4.1. **CMACP-init**$_{\text{UND}}$ *is PSPACE-complete.*

Proof. The membership of **CMACP-init**$_{\text{UND}}$ to PSPACE can be shown by using the same arguments as for the proof of Proposition 4.1. The proof of PSPACE-hardness is obtained by polynomial reduction from **CMAPF**$_{\text{UND}}$. We map a **CMAPF**$_{\text{UND}}$-instance (G, B, n, t) to the **CMACP-init**$_{\text{UND}}$-instance $(G', B, 2n, s)$ where G' is depicted in Figure 4.4 and starting at the configuration $s = \langle B, \ldots, B, v_1^1, \ldots, v_n^1 \rangle$. The definition of s means that n agents start at the base B and n agents start in positions $v_1^1, \ldots, v_n^1$, that are in the reduction gadget. Let us describe the construction of G'.

- *Nodes* We make four layers of vertices from $v_1^1, \ldots, v_n^1$ to $v_1^4, \ldots, v_n^4$.
- *Movement* We create movement edges between v_i^j and v_i^{j+1}, with $1 \leq i \leq n$ and $1 \leq j \leq 3$. Each node of the first and last layers have self-loops. The node v_1^4 has a movement edge to all nodes of G.
- *Communication* The node v_n^4 has a communication edge to all nodes of G and all nodes of the fourth layer. We create a communication edge between v_i^2 to t_i, for all $1 \leq i \leq n$. We create a communication edge between v_i^3 and v_{i+1}^3, for all $1 \leq i < n$. Finally, we connect in communication the nodes of the first layer and the node v_1^3 to the base B.

Formally, given (G, B, n, t) with $G = \langle V, E_m, E_c \rangle$ with base B, we define $(G', B, 2n, s)$ where $G' = \langle V', E'_m, E'_c \rangle$ with base B where:

– $V' := V \sqcup \{v_1^1, \ldots, v_n^1, v_1^2, \ldots, v_n^2, v_1^3, \ldots, v_n^3, v_1^4, \ldots v_n^4\}$

– E'_m is the symmetric closure of
$\cup E_m \cup \{(v, v_1^4) \mid v \in V\}$
$\cup \{(v_i^1, v_i^1), (v_i^1, v_i^2), (v_i^2, v_i^3), (v_i^3, v_i^4), (v_i^4, v_i^4) \mid i \in \{1, \ldots, n\}\}$

– E'_c is the symmetric closure of
$E_c \cup \{(B, v_1^1) \ldots (B, v_n^1)\} \cup \{(t_1, v_1^2), \ldots, (t_n, v_n^2)\}$
$\cup \left\{(B, v_1^3), (v_1^3, v_2^3), \ldots, (v_{n-1}^3, v_n^3)\right\}$
$\cup \left\{(v_1^4, v_n^4) \ldots (v_{n-1}^4, v_n^4)\right\} \cup \{(v_n^4, v) \mid v \in V\}$;

and $s = \langle B, \ldots, B, v_1^1, \ldots, v_n^1 \rangle$.

We now show that the instance (G, B, n, t) of **CMAPF**$_{\text{UND}}$ is feasible if and only if the instance $(G', B, 2n, s)$ of **CMACP-init**$_{\text{UND}}$ is feasible.

($\Rightarrow$) Suppose the configuration t is reachable in G. Let us construct an execution starting at s in G' that cover all nodes in G'. First, the first n agents reach configuration t, while the other n agents stay at layer v_i^1 using the self-loops. After that, the first n agents stay at t while the others progress to the fourth layer v_i^4. Finally, while the agents at v_2^4 to v_n^4 don't move using the self loops, the agent at v_1^4 covers the whole graph since the presence of an agent at v_n^4 makes sure that all nodes of G are connected to other agents and to the base. Once finished, this agent can go back to v_1^4 and the n agents can return back to the layer 1 as initially. Finally, the n agents occupying configuration t can go back to B by following the same path in reverse. This constitutes a covering execution for **CMACP-init**$_{\text{UND}}$.

($\Leftarrow$) Assume that G' can be covered, and let T denote the first time v_n^4 is visited. The agent at v_n^4 can only be at v_n^3 at time $T - 1$. Thus, the nodes $v_1^3, \ldots, v_{n-1}^3$ must also be occupied due

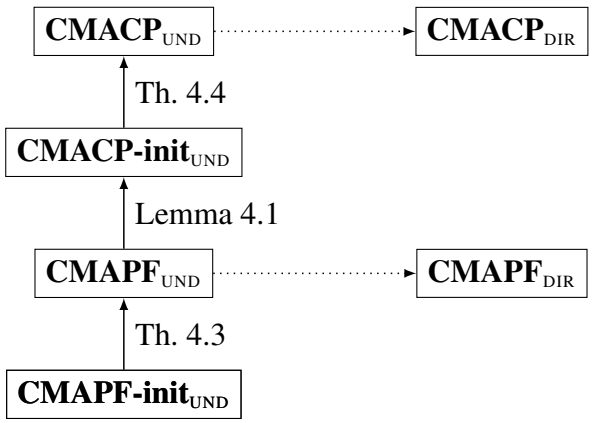


Figure 4.5 – Reductions of Section 4.4.

to connectivity constraint. Then, at time $T-2$, all n agents were on the second layer v_i^2. In fact, the layer 4 must be empty at this point by the choice of T and due to connectivity edges (and note also that there are no self-loops on the third layer). But since each node v_i^2 is only connected to t_i, at time $T-2$ the other n agent must be at configuration t. This concludes the proof. □

Theorem 4.4. **CMACP**$_{\text{UND}}$ *is PSPACE-complete.*

Proof. The upper bound comes from Proposition 4.1. We present the lower bound, which is by reduction from **CMACP-init**$_{\text{UND}}$. We map a **CMACP-init**$_{\text{UND}}$-instance (G, n, s) to the **CMACP**$_{\text{UND}}$-instance $(G', n+2)$ of where G' is defined as in Figure 4.3 (ignoring vertices $t_1, \ldots, t_n$). The formal description is given in the proof of Theorem 4.3.

We show that instance (G, n, s) of **CMACP-init**$_{\text{UND}}$ is feasible iff instance $(G', n+2)$ of **CMACP**$_{\text{UND}}$ is feasible. The proof is very similar to the proof of Theorem 4.3. The first n agents are used for the execution in G while the two others operate in the gadget.

($\Rightarrow$) If G can be covered starting from s then in G', the agents can first reach s as described in proof of Theorem 4.3, then follow the same plan to cover the graph, and execute the plan in reverse to come back to s and then back to B'. Meanwhile, the two others reach t_{n+1} and t_{n+2} loops there and come back to B'.

($\Leftarrow$) Assume there is a covering execution in G' from base B'. We already proved in Theorem 4.3 (Fact 1) that from configuration $\langle B', \ldots, B' \rangle$ the agents necessarily go through configuration $\langle s_1, \ldots, s_n, t_{n+1}, t_{n+2} \rangle$ to go in G. In the proof of Fact 1, observe that before t_{n+2} is first occupied no s_i can ever have been occupied, since these nodes are connected to B' only through t_{n+2}. Hence the next configuration where all s_i are occupied is the first visit of any agent to any s_i. Said otherwise, and generalizing to other instants when t_{n+2} starts being occupied, agents always enter G at $s_1, \ldots, s_n$ together. With a dual reasoning, one can see that always leave G together. Hence, the execution in G', minus the steps in the gadget, can be reproduced in G. □

We conclude this section by depicting, in Figure 4.5, the reductions used and the proof scheme. The dotted arrows represent the unmentioned corollaries.

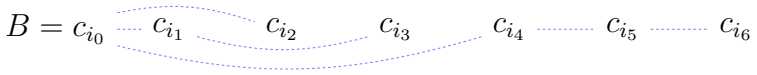


Figure 4.6 – Example of an ordering of nodes in $V' = \{c_1, \ldots, c_n, B\}$.

4.5 Sight-Moveable Topological Graphs

The main challenge in deciding whether there exists a connected plan is to verify that the given number of agents can visit a node of the graph while staying connected. Indeed, to visit a location connected to, say, the base, an agent might have to rely on multiple other agents. Hence, if we can guarantee that whenever two locations are connected we can move an agent from one to the other without the need of an extra relay, the problem becomes "easy". This assumption underlies the definition of sight-moveable topological graphs. Interestingly, the unbounded decision problems **CMAPF**$_{\text{SM}}$ and **CMACP**$_{\text{SM}}$ are in LOGSPACE and NLOGSPACE, respectively. Unfortunately, the bounded version **bCMAPF**$_{\text{SM}}$ is NP-complete. At the end of the section, we discuss a relaxation method based on this class of graphs.

4.5.1 Upper Bounds

Let us call USTCONN (resp. STCONN) the problem of determining whether two nodes of a given undirected (resp. directed) graph are connected. The algorithms presented in this section rely on the following complexity result:

Theorem 4.5 ([130])**.** *USTCONN is in LOGSPACE.*

Proposition 4.3. **CMAPF**$_{\text{SM}}$ *is in LOGSPACE.*

Proof. Let us define the problem UCONN as that of checking whether a given undirected graph is connected. By Theorem 4.5, this problem is in LOGSPACE since it suffices to check the connectivity between all pairs of nodes in LOGSPACE.

We are going to reduce **CMAPF**$_{\text{SM}}$ to UCONN in logarithmic space.

Let $G = \langle V, E_m, E_c \rangle$ a sight-moveable topological graph and c a configuration. Let $V' = \{c_1, \ldots, c_n, B\}$. We show that the configuration c is reachable iff the restriction of $\mathfrak{G}' := (V, E_c)$ to the nodes in V' is a connected graph. It is clear that this condition is necessary since if $\mathfrak{G}'$ is not connected then the agents cannot occupy configuration c.

Conversely, assume that $\mathfrak{G}'$ is connected. Then, let us order the nodes $B, c_1, \ldots, c_n$ into $c_{i_0}, c_{i_1}, c_{i_2}, \ldots, c_{i_n}$ with $c_{i_0} = B$, such that for all $1 \leq j \leq n$, c_{i_j} is connected to some c_{i_k} with $0 \leq k < j$; such an order exists, and can be obtained by breadth-first search in a E_c-spanning tree of V' at root B, see Figure 4.6 for an example.

We then construct an execution for reaching the configuration c as follows. The construction is by induction on $0 \leq j \leq n$: at step j, the first j agents occupy nodes $c_{i_1}, \ldots, c_{i_j}$. The base case $j = 0$ is the empty execution. For $j \geq 1$, let $k < j$ such that $(c_{i_k}, c_{i_j}) \in E_c$. We send the j-th agent to c_{i_k} following the path that has been constructed earlier, by induction. We then move that

agent to c_{i_j} following a path that stays connected with c_{i_j}, which exists by the sight-moveable property, the other agents either stay at c_{i_0} or at their respective c_{i_j} by using the self-loops.

Thus, (G, n, c) is a positive **CMAPF**$_{\text{SM}}$-instance iff $\mathfrak{G}'$ is a positive UCONN-instance. The reduction is in logarithmic space: we compute $\mathfrak{G}'$ by enumerating all E_c-edges (u, v) in G, and we output (u, v) when $u, v \in V'$. We recall that we only take into account the working memory for computing $\mathfrak{G}'$; the output – $\mathfrak{G}'$ itself – is not taken into account in the used space (see e.g. [143], Ch. 8, Def. 8.21). □

Before giving the complexity of **CMACP**$_{\text{SM}}$, we need the following intermediary result. Let us call Bounded-USTCONN the following problem: given an undirected graph $\mathfrak{G}$, two nodes s, t, an integer n, decide whether there is a path of length at most n from s to t in G. Note that, whatever the encoding of n is, the problem Bounded-USTCONN can be decided in logarithmic space.

Lemma 4.2. *Bounded-USTCONN is in NLOGSPACE.*

Proof. We reduce Bounded-USTCONN to STCONN in logarithmic space as follows. From a Bounded-USTCONN instance $(\mathfrak{G}, s, t, n)$ we construct in logarithmic space a STCONN instance $(\mathfrak{G}', s', t')$:

1. The nodes of $\mathfrak{G}'$ are pairs (v, j) where v is a node of $\mathfrak{G}$ and $j \in \{0, 1, \ldots, m\}$ where m is the minimum of n and the number of nodes of $\mathfrak{G}$.
2. The graph $\mathfrak{G}'$ contains an edge between (v, j) and $(v', j + 1)$ when there is an edge between v and v' in $\mathfrak{G}$ or when $v = v'$;
3. $s' = (s, 0)$ and $t' = (t, n)$.

The STCONN instance $(\mathfrak{G}', s', t')$ can be computed in log-space. Step 1 requires to store the current node v of $\mathfrak{G}'$ to be processed and an integer j written in binary, that is of size $\log m$. Importantly, as m is smaller than the number of nodes in $\mathfrak{G}'$, the representation of m is logarithmic in the size of the input. For this reason, note that the overall spatial complexity does not change even if the encoding of n is in binary. □

Proposition 4.4. **CMACP**$_{\text{SM}}$ *is in NLOGSPACE.*

Proof. Let $G = \langle V, E_m, E_c \rangle$ be a sight-moveable topological graph and n an integer written in unary. We prove that for all vertices v, there is a path of length at most n from v to the base c in the communication graph iff (G, n) is a positive instance of **CMACP**$_{\text{SM}}$. One direction is obvious: if (G, n) is a positive instance, then all vertices must within a distance of at most n from the base in the communication graph; otherwise no connected configuration can visit that vertex. Assume that all vertices are within a distance of n from the base. For any vertex v, consider path $v_1, \ldots, v_k$ in the communication graph, with $v_1 = c$, $v_k = c$, and such that $(v_i, v_{i+1}) \in E_c$. We apply the construction of Proposition 4.3 to build an execution from configuration c^n to some configuration where k agents occupy $\{v_1, \ldots, v_k\}$, and others stay at c. We now extend this execution to "roll back" so that all agents return to the base. We first let the agent at v_k, go to v_{k-1}: this is possible by the sight-moveable property and since $(v_k, v_{k-1}) \in E_c$. Then the two

agents together move to v_{k-2}, and so on, until all come back to c. We can now combine these executions to cover all vertices, and let agents end in the base.

Thus, the algorithm consists in checking sequentially, for all v, that $((V, E_c), v, c, n)$ is a positive instance of Bounded-USTCONN. Hence, we obtain a non-deterministic algorithm in logarithmic space to decide **CMACP**$_{\text{SM}}$. □

4.5.2 Lower Bounds

We now focus on the NP lower bound of **bCMAPF**$_{\text{SM}}$.

Theorem 4.6. **bCMAPF**$_{\text{SM}}$ *is NP-complete even for a fixed execution length* $\ell \geq 3$.

(a) Clause gadget

(b) Variable gadget

Figure 4.7 – Gadgets for reduction of **3-SAT** into **bCMAPF**$_{\text{SM}}$.

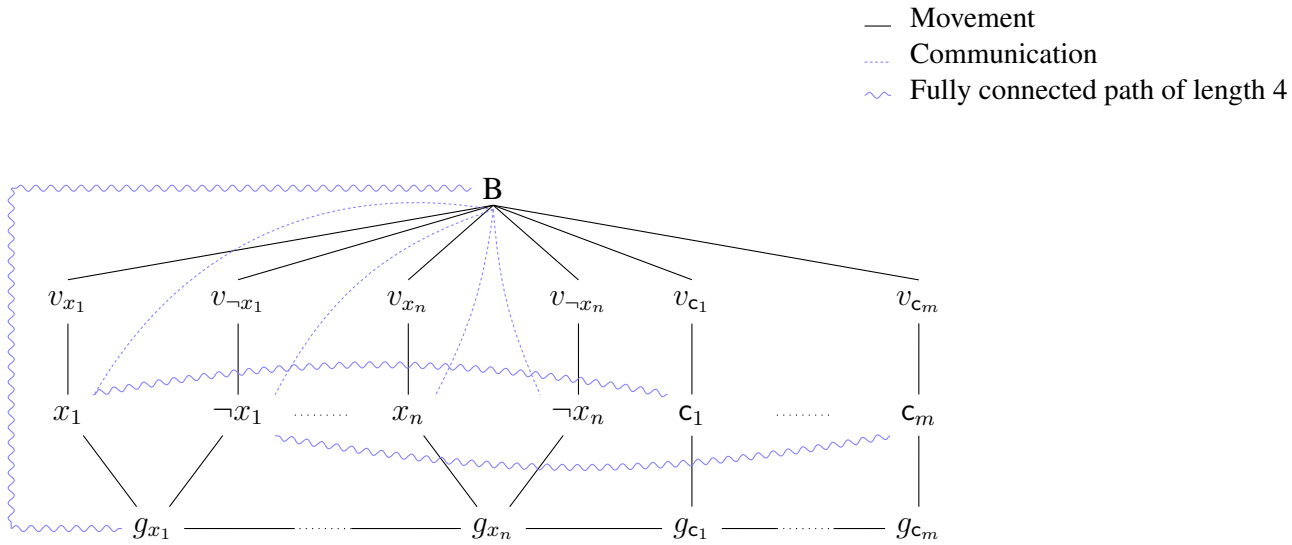


Figure 4.8 – Reduction of **3-SAT** into **bCMAPF**$_{\text{SM}}$. Communication edges implied by movement edges are not displayed. The variable x_1 is present in the clause c_1 and c_n.

Proof. The upper bound comes from Proposition 4.2. The lower bound proof is by polynomial time reduction from **3-SAT** problem (see [78]). Given a **3-SAT** instance, set of clauses $c_1, \ldots, c_m$ with variables $x_1, \ldots, x_n$, we describe the construction of an instance (G, k, c) of $\mathbf{bCMAPF}_{\text{SM}}$ with $k = n + m$ agents.

The topological graph $G = \langle V, E_m, E_c \rangle$ is constructed as follows. We start by placing the base B from which the agents start their mission.

Please recall that in a sight-moveable graph all movements edges have a respective communication edges, thus in the construction below we do not explicit them.

For each variable x, we construct a gadget composed of 5 nodes connected to the base depicted in Figure 4.7b: nodes x_i, $\neg x_i$, staging nodes v_{x_i}, $v_{\neg x_i}$ and a *goal* node g_{x_i}. We add movement edges from B to v_{x_i}, from v_{x_i} to x and from x to g_{x_i} (resp. from B to $v_{\neg x_i}$, from $v_{\neg x_i}$ to $\neg x_i$ and from $\neg x_i$ to g_{x_i}). As for the communication, the node x_i (res. $\neg x_i$) communicates with the base.

For each clause c_j, we construct a gadget composed of 3 nodes depicted in Figure 4.7a. We create a node c_j, a staging node v_{c_j} and a goal node g_{c_j}. We add movement edges from B to v_c, from v_{c_j} to c_j and from c_j to g_{c_j}. The communication between a clause c_j and a literal x_i or $\neg x_i$ is dictated by the existence of the literal in the clause. We do not use direct communication edges because the obtained topological graph should be sight-moveable. Instead, we use *fully connected paths of length 4*. Such a path between – let say – x_i and c_j consists three intermediate nodes p^1_{ij}, p^2_{ij}, p^3_{ij}, such that there is a path made up of movement edges from x_i to c_j, passing throw p^1_{ij}, p^2_{ij}, p^3_{ij}. Furthermore, we suppose that the nodes x_i, p^1_{ij}, p^2_{ij}, p^3_{ij} and c_j form a clique w.r.t. to the communication edges. Now, there is a fully connected path of length 4 between x_i and c_j if and only if $x_i \in \mathsf{c}_j$; and there is a fully connected path of length 4 between $\neg x_i$ and c_j if and only if $\neg x_i \in \mathsf{c}_j$.

We add movement edges from g_{x_i} to $g_{x_{i+1}}$, and from g_{c_j} to $g_{\mathsf{c}_{i+1}}$ for all $1 \leq i < n$, as well as from g_{x_n} to g_{c_1}. Last, we add a fully connected path of length 4 from g_{x_1} to the base such that $(g_{x_1}, B) \in E_c$, in the sense that all nodes of this path have communication edges between them. This translation is polynomial in the number of clauses and variables. The construction is depicted in Figure 4.8. The snake-like path from g_{x_1} to B is a fully connected path of length 4.

From a **3-SAT** instance, one can construct the graph G and ask for an execution of length 3 to reach the configuration $\langle g_{x_1}, \ldots, g_{x_n}, g_{\mathsf{c}_1}, \ldots, g_{\mathsf{c}_m} \rangle$.

Formally, the topological graph $G = \langle V, E_m, E_c \rangle$ is defined by:

– $V := \{B, v_{x_1}, v_{\neg x_1}, \ldots, v_{x_n}, v_{\neg x_n}, v_{\mathsf{c}_1}, \ldots, v_{\mathsf{c}_m}\}$
$\cup \{g_{x_1}, \ldots, g_{x_n}, g_{\mathsf{c}_1}, \ldots, g_{\mathsf{c}_m}\}$
$\cup \{p_B^1, p_B^2, p_B^3\} \cup \left\{p_{ij}^1, p_{ij}^2, p_{ij}^3 \mid i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}\right\}$

– $E_m$is the symmetric closure of
$\{(B, v_{x_1}), (B, v_{\neg x_1}), \ldots, (B, v_{x_n}), (B, v_{\neg x_n}), (B, v_{\mathsf{c}_1}), (B, v_{\mathsf{c}_1 x_m})\}$
$\cup \{(v_{x_1}, x_1), (v_{\neg x_1}, \neg x_1), \ldots (v_{x_n}, x_n), (v_{\neg x_n}, \neg x_n)\}$
$\cup \{(v_{\mathsf{c}_1}, \mathsf{c}_1), \ldots, (v_{\mathsf{c}_m}, \mathsf{c}_m)\}$
$\cup \{(x_1, g_{x_1}), (\neg x_1, g_{x_1}), \ldots, (x_n, g_{x_n}), (\neg x_n, g_{x_n})\}$
$\cup \{(\mathsf{c}_1, g_{\mathsf{c}_1}), \ldots, (\mathsf{c}_m, g_{\mathsf{c}_m})\}$
$\cup \left\{(g_{x_1}, g_{x_2}), \ldots (g_{x_{n-1}}, g_{x_n}), (g_{x_n}, g_{\mathsf{c}_1}), (g_{\mathsf{c}_1}, g_{\mathsf{c}_2}), ..., (g_{\mathsf{c}_{m-1}}, g_{\mathsf{c}_m}\right\}$
$\cup \{(B, p_B^1), (p_B^1, p_B^2), (p_B^2, p_B^3), (p_B^3, g_{x_1})\}$
$\cup \left\{(x_i, p_{ij}^1), (p_{ij}^1, p_{ij}^2), (p_{ij}^2, p_{ij}^3), (p_{ij}^3, \mathsf{c}_j) \mid x_i \in \mathsf{c}_j\right\}$
$\cup \left\{(\neg x_i, p_{ij}^1), (p_{ij}^1, p_{ij}^2), (p_{ij}^2, p_{ij}^3), (p_{ij}^3, \mathsf{c}_j) \mid \neg x_i \in \mathsf{c}_j\right\}$

– E_c is the symmetric and reflexive closure of
$E_m \cup \{(B, x_1), (B, \neg x_1), \ldots, (B, x_n), (B, \neg x_n)\}$
$\cup \left\{ \begin{array}{l} (B, p_B^1), (p_B^1, p_B^2), (p_B^2, p_B^3), (p_B^3, g_{x_1}), \\ (B, p_B^2), (p_B^1, p_B^3), (p_B^2, g_{x_1}) \\ (B, p_B^3), (p_B^1, g_{x_1}), (B, g_{x_1}) \end{array} \right\}$
$\cup \left\{ \begin{array}{ll} (x_i, p_{ij}^1), (p_{ij}^1, p_{ij}^2), (p_{ij}^2, p_{ij}^3), (p_{ij}^3, \mathsf{c}_j), & \\ (x_i, p_{ij}^2), (p_{ij}^1, p_{ij}^3), (p_{ij}^2, \mathsf{c}_j) & \mid x_i \in \mathsf{c}_j \\ (x_i, p_{ij}^3), (p_{ij}^1, \mathsf{c}_j), (x_i, \mathsf{c}_j) & \end{array} \right\}$
$\cup \left\{ \begin{array}{ll} (\neg x_i, p_{ij}^1), (p_{ij}^1, p_{ij}^2), (p_{ij}^2, p_{ij}^3), (p_{ij}^3, \mathsf{c}_j), & \\ (\neg x_i, p_{ij}^2), (p_{ij}^1, p_{ij}^3), (p_{ij}^2, \mathsf{c}_j) & \mid \neg x_i \in \mathsf{c}_j \\ (\neg x_i, p_{ij}^3), (p_{ij}^1, \mathsf{c}_j), (\neg x_i, \mathsf{c}_j) & \end{array} \right\};$

where nodes $p_\bullet^k$ are the intermediate nodes in the fully connected paths.

Fact 3. *G is a sight-moveable topological graph.*

Proof. Concerning the communication between the base B and the nodes x_i (resp. $\neg x_i$), a path does exist under the communication of B to reach x_i (resp. $\neg x_i$), due to communication induced by the movement. For the other communication edges, they are part of full connected paths which guarantee the sight-moveable condition. This ends the proof of Fact 3. △

Now let us prove that a **3-SAT** instance is satisfiable iff there exists an execution of at most 3 steps in the graph G.

($\Rightarrow$) We show that if a **3-SAT** instance is satisfiable then there exists an execution of at most 3 steps in the graph G built from it. Let val be a truth assignment which satisfies the instance. Recall that there are $n + m$ agents. The first step of the execution consists in moving an agent in each v_{c_i}, and for each variable x_i, moving one agent to v_{x_i} if the $val(x_i) = 1$ and to $v_{\neg x_i}$ otherwise. Note that all staging nodes communicate with B.

In the second step, all agents progress to their unique successors other than B. While all nodes x_i and $\neg x_i$ are connected to B, a node c_i is connected to B if and only if there is an

agent in one of its literals. This is the case since val satisfies the formula. In the third step of the execution, agents go to states g_{x_i} and g_{c_i}. Here, the connection with the base is ensured since g_{x_1} is connected to it, and g_{x_2} is connected to g_{x_1}, g_{x_3} is connected to g_{x_2} and so on.

This execution is thus a solution of **bCMAPF**$_{\text{SM}}$ with bound $\ell = 3$.

($\Leftarrow$) We show that if in the graph G there exists an execution of at most 3 steps constructed from a **3-SAT** instance, then the instance is satisfiable. Let us assume that we have an execution e of at most 3 steps with the last configuration being $\langle g_{x_1}, \ldots, g_{x_n}, g_{\mathsf{c}_1}, \ldots, g_{\mathsf{c}_m} \rangle$.

The only shortest path from B to g_{c_i} is of length 3 and goes through v_{c_i}. For states g_{x_i}, the only shortest paths are also of length 3 and go through either v_{x_i} or $v_{\neg x_i}$. Thus, in order to reach the given target configuration, at the initial step, agents must cover the states v_{c_i} and either v_{x_i} or $v_{\neg x_i}$ for all i, j. At the second step, following the above mentioned shortest paths, agents will be at states c_i and either x_i or $\neg x_i$ depending on the staging nodes they were occupying. The last step is the target configuration. Since the agents are connected at the second, it follows that for each clause c_j, the state corresponding to some literal of c_j is occupied by an agent. Thus, the valuation on variables encoded by the choices of the agents satisfies the **3-SAT** instance. □

4.5.3 Relaxation

For unbounded reachability and coverage, it seems tempting to take advantage of efficient algorithms on sight-moveable topological graphs (Proposition 4.3 and 4.4). In this subsection, we propose a transformation of a topological graph into a sight-moveable sub-graph. This transformation leads to a *relaxation* method: a solution found in the obtained sight-moveable topological graph still holds in the original graph.

This transformation requires the original graph to already satisfy properties 1 and 2 of Definition 4.3. If not, (1) we remove movement edges $(v, v') \in E_m$ that are not in E_c, (2) we remove all nodes without a self-loop.

Algorithm 1 Transformation into a sight-moveable topological graph

Require: A topological graph $G = \langle V, E_m, E_c \rangle$ satisfying properties 1 and 2 of Definition 4.3

$C := \emptyset$
for all $v \in V$ **do**
  $Q := \{v\}$
  while Q not empty **do**
    $v' := Q.pop()$
    for all $v'' \mid (v', v'') \in E_m$ **do**
      if $(v, v'') \in E_c$ **then**
        $C := C \cup \{(v, v'')\}$
        $Q := Q \cup \{v''\}$
$E'_c := \{(v, v') \mid (v, v') \in C \text{ and } (v', v) \in C \text{ or } v' = v\}$
return $\langle V, E_m, E'_c \rangle$

The simple transformation, given in Algorithm 1, prunes the communication edges which

do not respect the property 3 of sight-moveable topological graphs, described in Definition 4.3. We now show this transformation outputs a sight-moveable topological graph.

Theorem 4.7. *Algorithm 1 constructs a sight-moveable topological graph.*

For simplicity, we say that a node v is sight-moveable to a node v' iff there exists a sequence $\rho = \langle \rho_1, \ldots, \rho_n \rangle$ of nodes such that $v = \rho_1$, $v' = \rho_n$, $(v, \rho_i) \in E_c$ and $(\rho_i, \rho_{i+1}) \in E_m$ for all $i \in \{1, \ldots, n-1\}$. Algorithm 1 consists in running a breadth-first search from each vertex, and marking all reachable vertices v' to which v is sight-moveable. The communication edges are kept if the sight-moveable property was shown to hold in both directions.

Proof. First, given a node v, we show that the queue Q contains only nodes that v is sight-moveable to.

Invariant Given v, let I_v: for all $v' \in Q$, v is sight-moveable to v'.

Proof. Before entering the while loop, at Line 3, the queue Q contains only v. Hence, the Invariant I_v is initially satisfied. Let us suppose that Invariant I_v holds after an arbitrary number of loop iterations. If Q is empty, Invariant I_v holds. Otherwise, a node v' is popped out of Q. If a successor of v' communicates with v then we add the successors of v' to Q. Invariant I_v holds since v is sight-moveable to v', by induction, and $(v, v'') \in C$. △

We add a pair (v, v') to C iff v' was in Q and $(v, v') \in E_c$. Hence, given Invariant I_v, $(v, v') \in C$ iff v is sight-moveable to v'. Finally, E'_c is the set of pairs (v, v') such that $(v, v') \in C$ and $(v', v) \in C$, that is v is sight-moveable to v' and v' is sight-moveable to v. Therefore, the graph returned by Algorithm 1 is sight-moveable. □

This relaxation offers the opportunity to verify efficiently if a topological graph contains a solution. It is worth noting that the original topological graph might admit a solution, while no solution is found by the relaxation.

We illustrate the usage of the relaxation on the following maps, depicted in Figure 4.9, previously used by Tateo et al. [151], for the study of their algorithms. We use a similar discretization of the maps, i.e. the nodes of the graph are obtained by cells of 11×11 pixels for Office and 13×13 for Open, and we assume the communication range to be of 100 pixels. In addition, we use the same discretization with cells of 15×15 pixels for Coast, a map from the Benchmarks for Grid-Based Path-Finding [149], with an identical communication range.

We show, in Figure 4.10, the size of each component of the graphs obtained after discretization and the impact of the relaxation. The relaxation of Office removes up to 18% of the communication edges. We can observe that most of the communication through the walls, allowed by the communication range, will be removed in the relaxation since the sight-moveable property cannot be ensured. In particular the large room in the center loses a lot of communication edges with its surrounding. However, the relaxation of Open removes only 0.2% of the communication due to its small number of obstacles. Finally, the relaxation does not remove any communication edges from the discretization made of Coast.

Intuitively, the relaxation removes communication links between two vertices which cannot be connected by maintaining communication. In fact, if a node communicates through an obstacle, then the sight-moveable property requires it to have a bypass path that remains connected.

This is why, in the Office map, in particular at the borders of the central room, most communication links through walls are removed by the algorithm. But if the size of the obstacles is larger than the communication range, then there will be mostly no communication through obstacles, and it is unlikely that the relaxation will remove any communication edges: this can be observed on the Coast map where no edge is removed by the algorithm.

Thus, in open maps such as cities, forests etc., we expect that our relaxation removes a few edges and preserves the feasibility of our problems. It might however remove more communication edges, rendering the problems infeasible in indoor applications (inside of buildings, mazes, etc.). We leave the empirical evaluation of this relaxation for future work.

(a) Office

(b) Open

(c) Coast

Figure 4.9 – Maps.

Map/Graph	Original			Relaxation
	Nodes	Movement	Communication	Communication
Office	1669	5618	277059	227155
Open	2421	9114	295155	294503
Coast	5184	20448	580644	580644

Figure 4.10 – Characteristics of the discretized graphs and their relaxations.

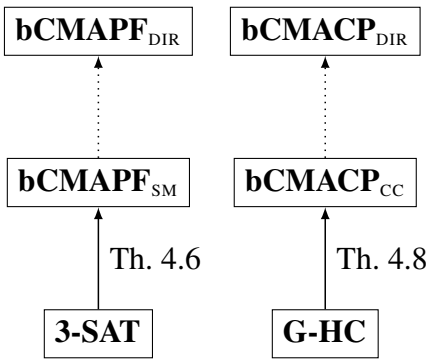


Figure 4.11 – Reductions of Sections 4.5 and 4.6.

4.6 Complete-Communication Topological Graphs

The following result relies on the fact that the communication is complete.

Proposition 4.5. **bCMAPF**$_{\text{CC}}$ *is in NLOGSPACE.*

Proof. We refer to Lemma 4.2. Indeed, given a configuration c and $\ell \in \mathbb{N}$, the straightforward iteration on the locations c_i followed by the verification of a path of at most ℓ (given in unary) steps from B to c_i yields a sound and complete algorithm for **bCMAPF**$_{\text{CC}}$. □

Our NP lower bound proof of the **bCMACP**$_{\text{CC}}$ problem is by reduction from the grid Hamiltonian cycle (**G-HC**) problem which is the Hamiltonian cycle problem restricted to grid graphs and is NP-complete [75]. We use this particular version of the Hamiltonian cycle problem for simplicity of the proof. Furthermore, we obtain a lower bound on the **bCMACP**$_{\text{CC}}$ problem with a grid movement graph.

Theorem 4.8. *Even restricted to grid graphs,* **bCMACP**$_{\text{CC}}$ *is NP-complete for a fixed number of agents* $n \geq 1$.

Proof. The upper bound follows from Proposition 4.2.

We give a polynomial-time reduction from the **G-HC** problem. Consider an instance of **G-HC** denoted $G = \langle V, E \rangle$.

Consider the sight-moveable topological graph $G' = \langle V, E_m, E_c \rangle$ with undirected movement $E_m = E$ and $E_c = V \times V$ and associate a single agent and the bound $|V|$ to the **bCMACP**$_{\text{CC}}$ instance. We call a simple cycle containing all vertices a *tour*. We prove that there exists a tour t in G iff there exists a covering execution of length $|V|$ in G'.

($\Rightarrow$) Any tour of G is a valid execution satisfying **bCMACP**$_{\text{CC}}$ since the communication edges form a complete graph, and the bound is $|V|$.

($\Leftarrow$) Let us suppose that we have an execution of length $|V|$ which covers the graph G'. The execution starts and ends at B and visits all nodes in $|V|$ steps. Hence, the execution visits all nodes only once and is a cycle in the graph. □

In Figure 4.11, we depicted the results obtained in this section and in Section 4.6.

4.7 Variants

In this section, we introduce several variants and study their impact on the complexity.

4.7.1 Bounded Reachability and Coverage with Binary Bounds

Our membership NP proofs for the bounded versions of the reachability and coverage problems rely on the unary encoding of the bound given in input. It is relevant to investigate the impact of providing that bound in binary on the complexity of the problems.

We show that both problems **bCMAPF**$_{\text{DIR}}$ and **bCMAPF**$_{\text{UND}}$ with the bound ℓ written in binary are both PSPACE-complete. The membership to PSPACE can be shown as in Proposition 4.1: since a binary counter can be added to count up to ℓ. For **bCMAPF**$_{\text{UND}}$, the PSPACE-hardness follows from a reduction from **CMAPF**$_{\text{UND}}$, whose PSPACE-hardness is established in Theorem 4.3. Indeed, any instance of **CMAPF**$_{\text{UND}}$ can be reduced to bounded reachability with bound $\ell = |V|^n$. In fact, if **CMAPF**$_{\text{UND}}$ has a solution, then there is a plan of length at most $|V|^n$. Indeed, in the worst case, the agents must go through all configurations possible in the graph. This number can be computed in polynomial time and represented in binary. The same argument is used to show the PSPACE-hardness of **bCMAPF**$_{\text{DIR}}$ by reduction from **CMAPF**$_{\text{DIR}}$ whose PSPACE-hardness was proved by Tateo et al. [151] (see Theorem 4.2).

Theorem 4.9. *The variants of* **bCMAPF**$_{\text{DIR}}$ *and* **bCMAPF**$_{\text{UND}}$ *with* ℓ *written in binary are PSPACE-complete.*

A similar argument can be used to show the PSPACE-hardness of the problems **bCMACP**$_{\text{UND}}$ and **bCMACP**$_{\text{DIR}}$. In fact, if there is a covering execution, then there must be one of length at most $|V|^n \times (|V| + 1)$. Indeed, the agents must traverse at most the $|V|^n$ possible configurations to reach each node of the graph and go back to the base, which means an upper bound of $|V|^n \times (|V| + 1)$ for the whole execution. This bound can be written in binary in polynomial space, so PSPACE-completeness also holds for these problems.

Theorem 4.10. *The variants of* **bCMACP**$_{\text{DIR}}$ *and* **bCMACP**$_{\text{UND}}$ *with* ℓ *written in binary are PSPACE-complete.*

4.7.2 Weighted Movement Graph

An interesting extension is obtained by assigning costs to edges of the movement graph, and considering the bounded reachability problem with respect to the total cost of the execution.

Consider a *weighted* topological graph $G = \langle V, E_m, E_c, cost \rangle$, where for each edge $cost(e)$ is the cost of e, a positive integer.

One could consider several ways of aggregating the weights along executions. We consider the case where the weights correspond to travel times between vertices, and the goal is to minimize total travel time, which also implies minimizing battery usage in drone applications. We consider a synchronous setting where all agents synchronize at each configuration in the plan; therefore, the travel time between two configurations is the maximum of the weights of the

edges along which agents travel. In other terms, we assume that agents wait for each other at each step of the execution.

Formally, for an execution $c^0, c^1, \ldots, c^k$, the total cost is defined as follows:

$$\sum_{i=0}^{k-1} \max_{j \in \{1,2,\ldots,n\}} cost(c_j^i, c_j^{i+1}), \tag{4.1}$$

where c_j^i denotes the vertex occupied by agent j at configuration c^i. Assuming binary encoding of all weights, we are interested in checking the existence of an execution of bounded cost, and that of a covering execution of bounded cost.

When weights are encoded in unary, both reachability and coverage problems are NP-hard since when all costs are equal to 1, then the problems are identical to **bCMAPF** and **bCMACP**, respectively. Since all weights are natural numbers, the length of the execution is not more than the cost bound given in input. Thus, similarly to Proposition 4.2, one can guess an execution and bound its length with a polynomial number of guesses.

Theorem 4.11. *The weighted variants of* **bCMAPF**, **bCMACP** *on directed and undirected topological graphs with unary encoding are NP-complete.*

When the weights are encoded in binary, then the problems are PSPACE-hard as shown in Section 4.7.1; and the PSPACE algorithms can be extended to establish PSPACE-completeness.

Theorem 4.12. *The weighted variants of* **bCMAPF**, **bCMACP** *on directed and undirected topological graphs with binary encoding are PSPACE-complete.*

4.7.3 Bounded Disconnection

Another extension consists in allowing the agents to be disconnected for a bounded number of steps along the execution. Such a feature can be desirable in order to allow the agents to reach difficult locations where communication cannot be guaranteed. By bounding the disconnection time, the connection with the base is only disturbed temporarily.

This extension is also PSPACE-complete. In fact, our membership results can easily be extended both for reachability and coverage problems. For PSPACE-hardness, observe that when the allowed bound is 0, the problems become identical to our setting, so all hardness proofs carry over.

4.7.4 Collisions

Collision constraints consist in disallowing several agents to share the same location at a given time, and provide an interesting extension of our setting. In our case, we allow several agents to be at the base but not at other nodes.

The PSPACE-completeness results hold by using the same reductions used for the proof of Theorems 4.3 and 4.4 from the collision-free variant problem of Tateo et al. [151]. Indeed, the reductions gadgets used are collision-free, and thus we obtain:

Theorem 4.13. *The variants with collision constraints of the problems* $\textbf{CMAPF}_{\text{UND}}$, $\textbf{CMAPF}_{\text{DIR}}$, $\textbf{CMACP}_{\text{UND}}$ *and* $\textbf{CMACP}_{\text{DIR}}$ *are PSPACE-complete.*

We can show that when the topological graph is sight-moveable, then the complexity of deciding the reachability and coverage without allowing collisions is unchanged. We use the same membership proof of Proposition 4.3 and 4.4. Indeed, a positive instance of $\textbf{CMAPF}_{\text{SM}}$ (resp. $\textbf{CMACP}_{\text{SM}}$) is a positive instance of $\textbf{CMAPF}_{\text{SM}}$ (resp. $\textbf{CMACP}_{\text{SM}}$) with collision constraints. In order to obtain a collision-free execution, it suffices to modify as follows: we consider an ordering, as depicted in Figure 4.6, and consider each branch one by one. For each branch, we dispatch first the agent with the furthest target from the base, followed by the second furthest agent and so on. This yields a collision-free execution for $\textbf{CMAPF}_{\text{SM}}$. $\textbf{CMACP}_{\text{SM}}$ can be solved by a repeated application of $\textbf{CMAPF}_{\text{SM}}$. Observe that $\textbf{CMAPF}_{\text{CC}}$ and $\textbf{CMACP}_{\text{CC}}$ can use the same algorithms.

Theorem 4.14. *The variants of the problems* $\textbf{CMAPF}_{\text{SM}}$, $\textbf{CMAPF}_{\text{CC}}$ *(resp.* $\textbf{CMACP}_{\text{SM}}$ *and* $\textbf{CMACP}_{\text{CC}}$*) with collision constraints are in LOGSPACE (resp. NLOGSPACE).*

Regarding the bounded versions, the membership of Proposition 4.2 hold without allowing collisions. Indeed, checking if the path contains a collision can be done in polynomial-time. The lower bound can be showed by reduction from the MAPF problem to $\textbf{bCMAPF}_{\text{CC}}$. Additionally, the proof of Theorem 4.8 holds without allowing collisions given that the proof holds for a single agent.

Theorem 4.15. *The variants of* $\textbf{bCMAPF}_{\star}$ *and* $\textbf{bCMACP}_{\star}$ *without collisions are NP-complete for all* $\star \in \{\text{UND}, \text{DIR}, \text{SM}, \text{CC}\}$.

4.7.5 Planar Movement Graphs

One could wonder whether the complexity of these problems change when restricted to topological graphs whose movement graphs are planar. This is an interesting question since in targeted applications the graphs are indeed planar; and some problems are known to become easier on planar graphs (e.g. the shortest path computation [80]). In this section, we show that our reductions proving the PSPACE-hardness of $\textbf{CMAPF}_{\text{UND}}$, $\textbf{CMACP-init}_{\text{UND}}$, $\textbf{CMACP}_{\text{UND}}$ and $\textbf{CMAPF}_{\text{SM}}$ use planar graphs, which means that these hardness results hold even when the input is restricted to planar graphs.

Actually, the proof of $\textbf{CMAPF-init}_{\text{UND}}$, by Tateo et al. [151], uses a planar movement graph. They even prove a stronger result: the reachability problem is PSPACE-hard even for graphs that are disjoint unions of *paths* [2].

Let us consider an instance $(G_{Reach_{init}}, B, s, t)$ of $\textbf{CMAPF-init}_{\text{UND}}$ where $G_{Reach_{init}}$ is a topological graph that is a disjoint union of paths (depicted in the middle of Figure 4.12), B is

2. Their reduction is from a problem called *the reconfiguration problem on constraint graphs*. The topological graph computed from a constraint graph contains single nodes (○), paths of length 3 (which we denote by ○—■—○) and a single path graph of length $|E|$ (denoted by ■——— ··· ———■), with E the set of edges in the constraint graph.

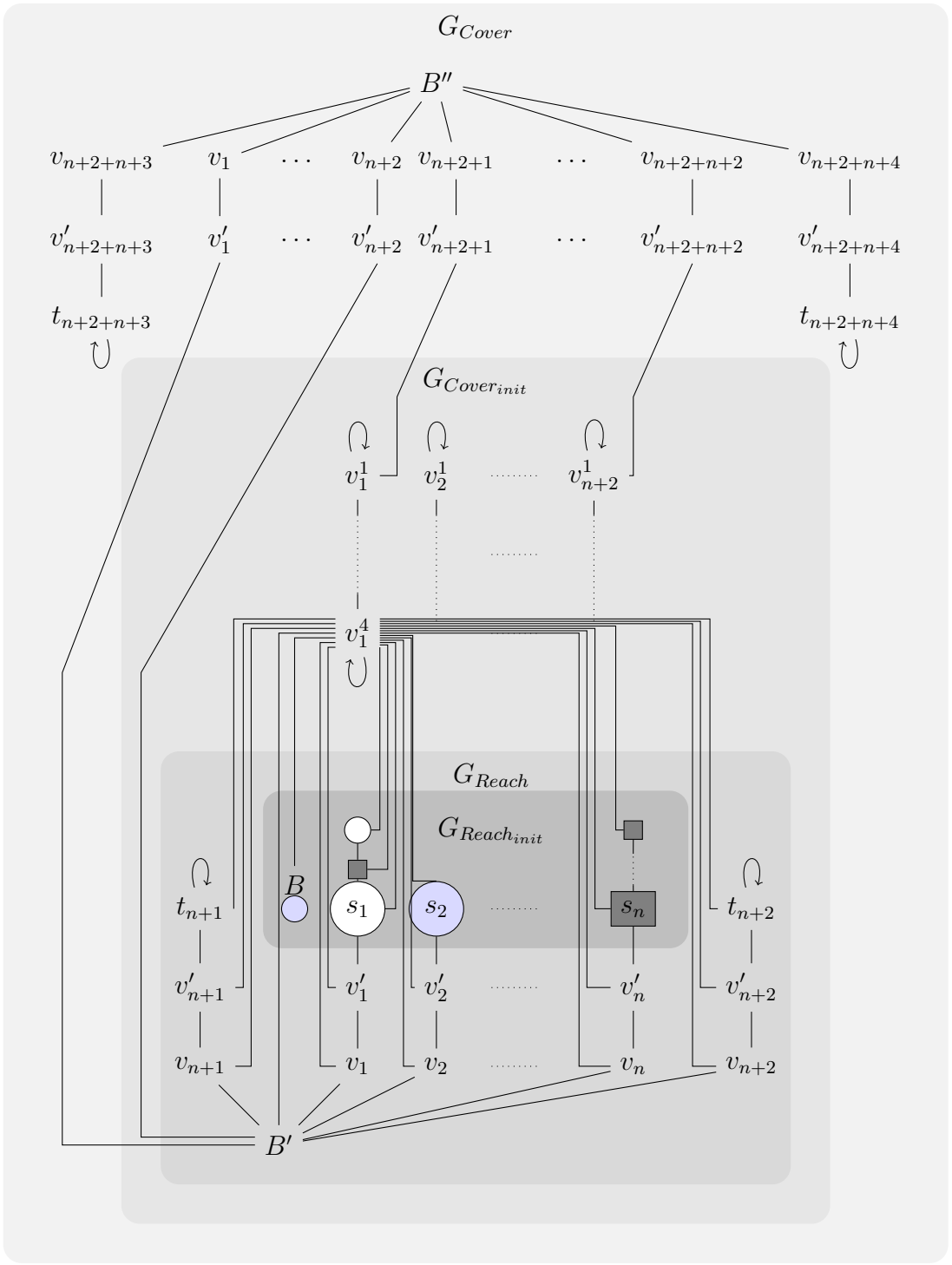


Figure 4.12 – Summary of all the planar constructions.

its base, $s = (s_1, \ldots, s_n)$ is initial configuration for n agents, depicted at the bottom of $G_{Reach_{init}}$ in Figure 4.12 and $t = (t_1, \ldots, t_n)$ is the target configuration (the target nodes $t_1, \ldots, t_n$ are in $G_{Reach_{init}}$ as defined by Tateo et al. [151] but are not displayed in the figure).

Then as shown in Figure 4.12, the gadgets we proposed in this contribution can all be arranged to obtain planar graphs as well.

1. The gadget given in Figure 4.3, used in the proof of Theorem 4.3 is a planar graph; in fact, it can be arranged, to obtain G_{Reach} (in Fig. 4.12) whose base is B' and is designed for $n + 2$ agents. This proves that **CMAPF**$_{\text{UND}}$ is PSPACE-hard on planar graphs.
2. We consider now the reduction of Figure 4.4 used in proof of Lemma 4.1, which contains G_{Reach}. This reduction, named $G_{\text{Cover}_{\text{init}}}$ in Figure 4.12 is also planar. The base of this graph is still B', and it is designed for $n + 2 + n + 2 = 2n + 4$ agents. In this reduction, the first $n + 2$ agents start in B' while the others start in $v^1_1, \ldots, v^1_{n+2}$. Indeed, the latter gadget is disconnected from the rest of the graph, except that the node v^4_1 has a movement edge to all nodes in the graph. One can observe that a movement edge can be created from v^4_1 to all nodes in the graph G_{Reach}, while keeping the construction planar, thanks to the particular structure of G_{Reach}. This shows that **CMACP-init**$_{\text{UND}}$ is PSPACE-hard on planar graphs.
3. Finally, we can add the gadget of Figure 4.3 used in proof of Theorem 4.4, shown in the top part of Figure 4.12. We thus obtain G_{Cover} designed for $2n + 4 + 2 = 2n + 6$ agents starting at the base B'', which is the reduction of Theorem 4.4. The obtained graph is also planar. This shows that **CMACP**$_{\text{UND}}$ is PSPACE-hard on planar graphs.

To sum up:

Theorem 4.16. **CMAPF**$_{\text{UND}}$, **CMACP-init**$_{\text{UND}}$ *and* **CMACP**$_{\text{UND}}$ *are PSPACE-hard on planar graphs.*

4.8 Conclusion

We have studied numerous variants of the defined problems in the hope to give a complete overview of this new extension of the MAPF problem. We provided a thorough study of coordination problems under connectivity constraints for reachability and coverage. This study carves an initial map of the complexity and the difficulties that can rise while solving such problems.

We still lack results for some of these variants. Furthermore, it is also unclear whether the combination of these variants can yield higher complexity. Additionally, there are still many variants of MAPF that have not been extended to these problems.

An important result of this chapter is the introduction of sight-moveable topological graphs. Indeed, being able to decide in NLOGSPACE whether the reachability or the coverage can be done is an important improvement over the previous results. However, the optimization of the execution cannot be solved in a easier manner in this class.

In realistic situation, the topological graph may not be fully known in advance. Indeed, the discretization may not represent faithfully the area in which the agents evolve. Thus, we

attend to extend this work to incomplete knowledge in which the agents only have an over-approximation of the actual topological graph. While the agents move around the area they can observe the graph and update their knowledge.

Finally, not only can the area be partially known by the agents, but some mishaps can happen during the mission. Indeed, an external actor might disable an agent and, in this setting, the group of agent should have to find the agent or finish the mission without it. In the weighted graph extension (Subsection 4.7.2), mishaps can also be due to strong wind which modifies the cost of the movement.

CHAPTER 5

CONNECTED-CONFLICT-BASED SEARCH

5.1 Introduction

In this chapter, we study an optimal decoupled algorithm for CMAPF that can be extended to the collision-avoidance requirement. Our algorithm is an adaptation of CBS [137, 139] to handle connectivity constraints. We thus present the first optimal algorithm that handles connectivity in this setting.

Both MAPF and CMAPF could be solved offline as a single-agent search with A* [64]. However, this would be exponential in the number of agents and available moves. This *coupled* approach is computationally intensive and less effective, in general, than a *decoupled* approach in which paths of the agents are computed separately. This is the approach taken by CBS to solve MAPF. CBS computes optimal paths for each agent separately; and when a collision between two agents is detected at a location at a given time step, it enforces one of the agents away from that location at that time. Numerous improvements have been introduced to CBS [20, 46, 90, 91, 92].

Our contribution is an optimal algorithm called *Connectivity-Conflict-Based Search (CCBS)* to solve CMAPF. The main structure of the algorithm is based on CBS with new enhancements to deal with the particular challenges of CMAPF. Observe that CBS deals with collision conflicts between two agents by enforcing one of the agents to move away from the location of the collision. In CCBS, *connectivity conflicts* correspond to disconnected configurations, that is, configurations where the agents do not form a connected subgraph. Resolving such a conflict is more tricky: a connectivity conflict can involve several agents (i.e. one may need to relocate several agents to obtain a connected configuration) and a single disconnected agent may not be the source of the conflict. Thus, unlike CBS, our algorithm may need several steps to resolve a connectivity conflict.

Outline Section 5.2 presents our algorithm. Section 5.3 contains the correctness and optimality proofs. Section 5.4 presents experiments, and we conclude and discuss the many possible extensions of this work in Section 5.5.

5.2 Connected-Conflict-Based Search

In this section we describe our algorithm called Connectivity-Conflict-Based Search (CCBS) for CMAPF. Let us first introduce an example used to illustrate some notions during the presentation of our algorithm.

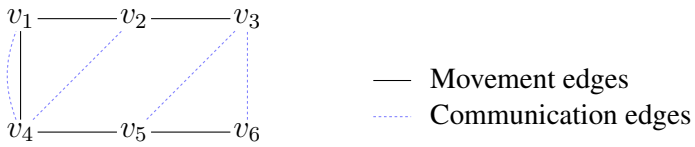


Figure 5.1 – An example of a topological graph.

Example 5.1. *Consider the topological graph of Figure 5.1. Consider the instance of CMAPF with $k = 2$ agents with starting configuration $s = \langle v_1, v_4 \rangle$ and goal configuration $g = \langle v_3, v_6 \rangle$. The execution $([v_1, v_2, v_3], [v_4, v_5, v_6])$ is not connected. Indeed, at the second step the configuration is $\langle v_2, v_5 \rangle$: no communication edge exists between v_2 and v_5. However, the execution $([v_1, v_2, v_3, v_3], [v_4, v_4, v_5, v_6])$ is connected since the configurations $\langle v_1, v_4 \rangle$, $\langle v_2, v_4 \rangle$, $\langle v_3, v_5 \rangle$ and $\langle v_3, v_6 \rangle$ are connected. This execution is actually an optimal solution of this CMAPF instance and is of cost 4.*

Our algorithm is inspired by the general concept of CBS [139] which solves MAPF by finding an optimal collision-free execution (without any connectivity constraints). CBS solves the MAPF problem in a *decoupled* manner by computing separately the path of each agent and progressively repairing the resulting execution by constraining agents away from collision locations. CBS is correct and optimal, that is, it finds a collision-free execution with minimum cost whenever there is at least a solution, and, as shown by Sharon et al. [139], it can be made complete by the addition of a polynomial procedure [172], that is, it can also detect the absence of solution.

Similarly to CBS, CCBS is composed of two levels, the *high-level* constructs a *constraint tree* which searches for the right set of constraints to add; and the *low-level* computes single-agent paths satisfying given constraints.

Definition 5.1 (Constraint)**.** *A positive constraint is a tuple denoted by $\langle a, v, t \rangle$ where a is an agent, v a vertex, and t a time step.*

Intuitively, a positive constraint $\langle a, v, t \rangle$ means that agent a *must* be at vertex v at time step t. Formally, given an execution exec, the path exec_a for agent a is *consistent* with such a constraint if $\text{exec}_a[t] = v$. An execution exec is consistent with a set of constraints if all paths exec_a are consistent with all constraints in the set.

Notice the difference with CBS [139] which only uses negative constraints, while we use positive ones. Positive constraints were already also used by Li et al. [91] for collision constraints only.

We will say that an execution exec has a *connectivity conflict* at time t, if there exists t such that $\text{exec}[t]$ is a disconnected configuration.

5.2.1 The High-Level: The Constraint Tree

Given an instance of CMAPF $\langle G, k, s, g \rangle$, the high-level of CCBS builds a *constraint tree* containing constraint nodes.

Definition 5.2 (Constraint Node)**.** *A constraint node* n *is composed of the following attributes:*

1. n.cons — *a finite set of constraints;*
2. n.exec — *an execution;*
3. n.cost — *the cost of the execution.*

Initially, the tree contains a single root node n with an empty set of constraints, an execution composed of a shortest path from s_a to g_a for each agent a.

If the execution is *connected*, we return it. Otherwise, we say that the execution contains connectivity conflicts, that is, disconnected configurations. In this case, we *split* this constraint node as follows: we select (1) a disconnected configuration at a timestep t, (2) and an agent a, and (3) for each vertex $v \in V$, we create a successor constraint node n' with constraints $\mathsf{n}.\text{cons} \cup \{\langle a, v, t\rangle\}$; we assign a shortest path for a satisfying these constraints, keep other agents' paths unchanged, and compute the cost of this new solution.

These generated constraint nodes are stored in a priority queue, called OPEN, ordered by the cost of their executions, and we break ties by choosing nodes with the least number of connectivity conflicts. Note that a child node added for a connectivity conflict at time step t may still contain a connectivity conflict at time step t; in fact, one may need to add several constraints (that is, to move several agents) in order to render the configuration connected. When generating children nodes for a connectivity conflict at time t, we thus do not consider agents for which there is already a constraint at time t since this would be either redundant or make the node unsatisfiable.

Note that for each connectivity conflict at time t and an agent a, one of the generated child node constrains agent a to his current location (this is because a child node is created for *all* $v \in V$). This allows us to guarantee the optimality of the algorithm; in fact, if the optimal solution requires the agent to remain at his current position, we *do* keep a branch in the constraint tree consistent with an optimal solution.

We describe the 3 main steps of the high-level algorithm:

(1) Choice of the connectivity conflict Once a constraint node is selected for splitting from the priority queue, we need to select a connectivity conflict, that is, a disconnected configuration in the execution. An execution contains multiple conflicts in general, and the order in which these are addressed can have an impact on the performance of the search.

Notice that one may need to modify several paths, that is, add several constraints, in order to render a configuration connected. Each such constraint corresponds to a level in the constraint tree, so several levels may be required to transform a single configuration into a connected one. Intuitively, we would like the search to avoid exploring too deep in the constraint tree too quickly, so we prefer conflicts that require a small number of constraints. We thus select a conflict with the least number of connected components in the communication graph of the configuration. The constraints added to solve such a conflict of course impact other time steps as well. Although these might create new conflicts, they can also solve other conflicts; intuitively, the hope is that the constraints often force the agents to remain close to each other rather than drift apart thus often guide us towards a solution. We empirically observed that this choice improved the performance.

(2) Choice of the agent In the worst case, one might need to relocate almost all agents in order to render a configuration connected. However, this is often not the case. In order to solve a connectivity conflict with a small number of constraints (thus, a small number of levels in the constraint tree), we select an agent in one of the smallest connected components of the communication graph of the configuration. For instance, an agent which seems to be alone and disconnected from the rest of a big group is selected and relocated first.

(3) Reducing the branching factor When splitting a constraint node, we generate a new constraint node for each available position of the selected agent, generating a total of $|V|$ children nodes. Such a branching factor is not practical for an efficient search. Fortunately, most of the positions are often not reachable by the agent at the considered time step given the set of constraints of the node. Hence, the high-level algorithm creates a large number of nodes which are not satisfiable. In order to limit the generation of such nodes, we consider the following two optimizations.

Partial Splitting: The idea of this optimization is that when splitting a node n, we give priority to child nodes that have a better chance of not extending the execution. In fact, given a conflict at time t and agent a, we first only generate successor nodes for each v reachable by a in exactly t steps; and put the parent node n back to OPEN with its cost increased by 1. The next time the node n is selected for splitting, we continue generating nodes for vertices v reachable by a in exactly $t-1$ steps etc. Notice that the second set of nodes will necessarily increase the cost since they constrain the agent, at time t, to a vertex that they could reach in $t-1$ steps (while this might be necessary to ensure connectivity). Observe also that a constraint with v only reachable with more than t steps would be unsatisfiable so these are not generated. This optimization is inspired by Enhanced Partial Expansion A* (EPEA*) [45].

Future constraint: When an agent a is constrained to be at location v at time t, this constrains the first portion of the path to be of length exactly t, and to end in v. Thanks to this simple observation, when addressing a connectivity conflict at time $t' < t$, we can consider only those constraining at locations v' at time t' such that there exists a path of length t' from s_a to v', and a path of length $t - t'$ from v' to v.

Both optimizations can be implemented using *multi-value decision diagrams* (MDDs) [20]. Indeed, given agent a and cost ℓ, MDDs allow one to build a compact representation of all paths of cost ℓ for agent a. An MDD has a *level* for each time step, and level t contains all locations at which the agent a can be at time t along paths that reach the goal location with cost exactly ℓ. In particular, if agent a can reach their goal with cost ℓ, then level 0 only contains the start location and level ℓ only contains the goal location. We denote by $\mathsf{MDD}_a^\ell[t]$ the set of all locations reachable by agent a with cost t along paths of cost ℓ that end in the goal. Figure 5.2a (resp. 5.2b) is the MMD of cost 2 (resp. 3) of agent 1 in Example 5.1. Note that MDD_1^3 does not contain the path $[v_1, v_2, v_3, v_3]$, as idling on the goal has cost 0, so this path has cost 2.

The overall algorithm is given in Algorithm 2, which shows how the constraint tree is created. It maintains a priority queue OPEN which stores the set of leaf nodes that have not been expanded yet. It runs as long as a solution has not been found and OPEN is non-empty. At each iteration, the best node is picked (Line 3). If this node has no connectivity conflict, it means that a solution is found (Line 4). If the node has connectivity conflicts, then such a conflict

(a) MDD_1^2 (b) MDD_1^3

Figure 5.2 – MDDs for agent 1 in Example 5.1.

Algorithm 2 High-Level of CCBS

Require: Topological graph $G = \langle V, E_m, E_c \rangle$, configurations s, g (considered as global variables)

```
1: INSERTROOT
2: while OPEN is not empty do
3:   n := remove the best node from OPEN
4:   if n has no conflict then return n.exec
5:   CHILDREN := empty list
6:   t := time-step of the chosen conflict in n.exec (1)
7:   a := agent to split at t in n (2)
8:   SPLIT(n, t, a)
9:   if BYPASS was raised then
10:    discard CHILDREN and insert n again into OPEN.
11:  else
12:    insert all nodes in CHILDREN into OPEN
```

Algorithm 3 Sub-procedures

```
1: procedure INSERTROOT
2:   root := new node
3:   root.cons := ∅
4:   for all agents a do root.exec_a := CSP(s_a, g_a, ∅)
5:   root.cost := Cost(exec)
6:   insert root into OPEN

7: procedure CREATECHILD(node n, constraint ⟨a, v, t⟩)
8:   n' := new node with n'.cons := n.cons ∪ {⟨a, v, t⟩}
9:   for all agents b ≠ a do n'.exec_b := n.exec_b
10:  n'.exec_a := CSP(s_a, g_a, n'.cons)
11:  n'.cost := Cost(exec)
12:  if n'.cost = n.cost and n' has less conflicts than n then
13:    n.exec_a := n'.exec_a
14:    raise BYPASS
15:  insert n' into CHILDREN

16: procedure SPLIT(node n, time-step t, agent a)
17:  for all v ∈ V and following (3) do
18:    CREATECHILD(n, ⟨a, v, t⟩)
```

and an agent are chosen and child nodes are created (Lines 6 and 7). The last part of the algorithm shows the bypass optimization from Boyarski et al. [19] which is the following. If a child node n' of n has the same cost as n, but has fewer conflicts, then we replace the execution of n by that of n', discard all children, and put n back in OPEN. This allows us to bound the size of the constraint tree.

5.2.2 The Low-Level: Constrained Shortest Paths

We use an algorithm similar to the one described by Li et al. [91] to compute the constrained shortest paths for individual agents. Given a set of positive constraints, a start and a goal vertex, we use the positive constraints as timely ordered *landmarks*. We compute a shortest path from the start location to the first landmark using the original low-level of CBS [137] (time-space A*). Then from the first landmark to the second and so on until the goal vertex. No time bound is put on the path to the goal, while one is used on landmarks, which correspond to the times given in the constraints. Remark that if a landmark cannot be reached in the given time then there is no path satisfying the constraints.

Algorithm 3 shows the sub-procedures used in CCBS. CSP is a call to the low-level, that is, $\text{CSP}(s_a, g_a, c)$ returns a shortest path from s_a to g_a respecting the constraint set c.

5.3 Theoretical Analysis

In this section, we discuss the theoretical analysis of CCBS. Both following theorems are similar to the proofs of CBS [139].

Lemma 5.1. *At any moment during the execution of the algorithm, for all connected executions* exec *from s to g, there is a node* n *in* OPEN *that is consistent with* exec.

Proof. Consider such an execution exec. We prove the lemma by induction on the expansion of the search tree in OPEN. At the beginning, the root does not contain any constraints, thus exec is consistent with the root node.

Let us assume that the property is true at step $i-1$, and let us denote the content of OPEN by $\mathsf{n}_1, \ldots, \mathsf{n}_k$. At the i-th step of expansion, we choose a node, let it be n_1. If one of the $\mathsf{n}_2, \ldots, \mathsf{n}_k$ is consistent with exec, then we are done. Otherwise, by induction, n_1 is consistent with exec. Moreover, if n_1 does not have conflicts, we are also done.

Otherwise, assume first that n_1 contains a connectivity conflict. The algorithm creates a child node for each possible position of an agent a at a time t. Hence, a child node is created with the additional constraint $\langle a, \text{exec}[t], t\rangle$ and added to OPEN. This node is thus consistent with exec. Last, notice that if the bypass optimization is applied, then the node n_1 is put back in OPEN. □

Lemma 5.2. *At any moment during the execution of the algorithm, for all constraint nodes* n*,* $\mathsf{n}.\text{cost} \leq \text{cost}(\text{exec}^*(\mathsf{n}))$ *with* $\text{exec}^*(\mathsf{n})$ *an optimal execution consistent with* n.

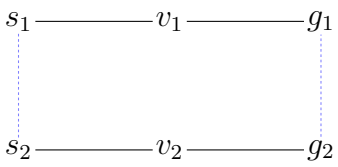


Figure 5.3 – An example of a topological graph with the negative instance $s = \langle s_1, s_2 \rangle$ with goal $g = \langle g_1, g_2 \rangle$.

Proof. The cost $\mathsf{n}.\mathrm{cost}$ is the sum of the cost of the *shortest* paths of each agent consistent with their respective constraints in $\mathsf{n}.\mathrm{cons}$. Since the path of each agent in the execution $\mathrm{exec}^*(\mathsf{n})$ also satisfies all constraints $\mathsf{n}.\mathrm{cons}$, the inequality follows. □

We are now ready to prove the soundness and completeness theorems.

Theorem 5.1 (Soundness)**.** *Any solution returned by CCBS is optimal.*

Proof. Let exec be an optimal solution. The algorithm returns a solution if it pops a node n from OPEN which contains an execution without conflicts. By Lemma 5.2, we know that $\mathsf{n}.\mathrm{cost} \leq \mathrm{cost}(\mathrm{exec}^*(\mathsf{n}))$, and since $\mathsf{n}.\mathrm{exec}$ has no conflicts, we have $\mathsf{n}.\mathrm{cost} = \mathrm{cost}(\mathrm{exec}^*(\mathsf{n}))$. Furthermore, Lemma 5.1 shows that exec is consistent with some node n' in OPEN at this moment, which means that $\mathsf{n}'.\mathrm{cost} \leq \mathrm{cost}(\mathrm{exec})$. But since the algorithm picks the node with the least cost, we also have $\mathsf{n}.\mathrm{cost} \leq \mathsf{n}'.\mathrm{cost}$. It follows that $\mathsf{n}.\mathrm{cost} \leq \mathrm{cost}(\mathrm{exec})$, so the returned solution $\mathsf{n}.\mathrm{exec}$ is optimal. □

Theorem 5.2 (Completeness)**.** *If there is a solution, CCBS returns a solution.*

Proof. Notice first that OPEN never becomes empty until a solution is found by Lemma 5.1. Assume that there is a non-terminating execution, thus generating an infinite constraint tree.

We claim that the cost of any leaf at a depth m is at least $\frac{m}{k|V|}$ where k is the number of agents. In fact, by construction, there can be at most one constraint per agent and time step, so if $t_{\max}$ denotes the maximal time step that appears in all the constraints along the branch up to depth m, then we have $m \leq k|V|t_{\max}$, that is $t_{\max} \geq \frac{m}{k|V|}$. So the path of some agent has length at least $t_{\max}$. Now consider the node n at which a constraint at $t_{\max}$ appears. Since goal configuration g is assumed to be connected, $\mathsf{n}.\mathrm{exec}[t_{\max}]$ must be different from g. Thus, the path of some agent a has length more than $t_{\max}$. Since this path is a shortest path satisfying the constraints $\mathsf{n}.\mathrm{cons}$, the path of agent a at any leaf node below n cannot be shorter. Thus cost at all such leafs is at least $\frac{m}{k|V|}$.

In other terms, the cost of each branch diverges to infinity as the branch is extended. This implies that each leaf node is eventually split since it will eventually become the top node in the priority queue. So, at some point, the queue only contains nodes of costs strictly greater than the cost of exec^*. By Lemma 5.1, one node n of these nodes is consistent with exec^*. We thus have $\mathsf{n}.\mathrm{cost} > \mathrm{cost}(\mathrm{exec}^*)$. By definition, we have $\mathrm{cost}(\mathrm{exec}^*) \geq \mathrm{cost}(\mathrm{exec}^*(\mathsf{n}))$. Contradiction with Lemma 5.2. □

If there is a solution, CCBS returns an optimal solution. However, if there is no solution, CCBS runs forever, as does CBS. For example, consider the instance with two agents given in Figure 5.3. One can observe that there is no connected execution for the two agents. The constraint tree will grow infinitely enforcing the agents to stay at their starting locations.

As done by Sharon et al. [139], we may use an algorithm to check the existence of a solution before running CCBS. While this check is in polynomial time in the collision-only case [172], in presence of connectivity constraints, checking the existence of a solution is PSPACE-complete [151].

5.4 Experimental Results

In this section, we evaluate the performance of our algorithm and compare it with the two algorithms of Tateo et al. [151], namely, the Sample-Based (SB) and Depth-First Search (DFS) algorithms.

The former algorithm creates an execution by, starting from the initial configuration, exploring only a subset of all the possible connected steps by performing random movements. A heuristic then evaluates the "best" state. Then the process is iterated until the "best" state is the goal state. The latter algorithm is, intuitively, similar to iterations of A* of depth 1. More precisely, the algorithm, starting from the initial configuration, searches the "next best connected state", which minimize a heuristic. From this state the process is iterated until it reaches the goal.

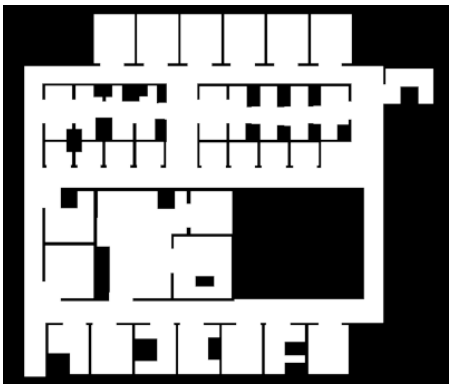
The experiments were carried out on 2 benchmark maps, depicted in experimental analysis of Hollinger and Singh [68] and Tateo et al. [151]. *Offices* is a map of the SDR offices from the Radish data set [72]. *Open* is a map of the McKenna MOUT site. The movement edges follow a 4-way grid (i.e. the agents can move in the 4 directions). Concerning communications, we adopted a distance-based one. In *distance-based* communication, an agent communicates with all other agents within a maximal distance, called the range; the range is displayed below the maps in Fig. 5.4.

All algorithms were implemented in C++ and run on an Intel i7 5600U 2.60 GHz (3.20 GHz) with 8 GB of available memory (out of 16 GB). We used the open source programs published by [151] to run their algorithms (which are also in C++). For both maps and for each number n of agents, with $n \in \{2, 3, \ldots, 20\}$, we had 100 instances, and we gave each algorithm 1 minute per instance. The success rate of an algorithm on these benchmarks is defined as the percentage of instances for which it found an execution in the allocated time. Note that CCBS is an optimal algorithm unlike all other algorithms.

The success rate of the algorithms are given in Figure 5.5 for both class of benchmarks. Figure 5.6 depicts the average size of the execution generated by the algorithm.

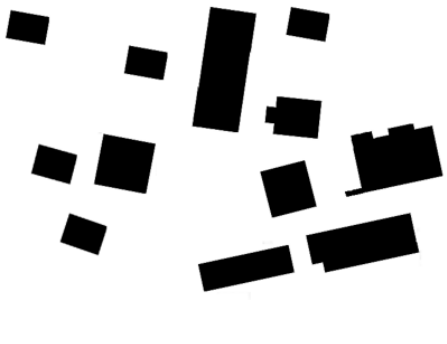
The two considered maps are of different nature. Open has fewer obstacles, which are moreover thick (these correspond to buildings), and it has many communication edges; Offices has many thin obstacles through which agents cannot move but can communicate.

Instances on Open seems to be easier to solve since all algorithms achieve a higher success rate for a few agents; while CCBS scales better: the success rate of DFS and SB drop



#nodes	1494
#mvt edges	2419
#comm edges	29962

(a) Offices Map.



#nodes	2205
#mvt edges	4107
#comm edges	39310

(b) Open Map.

Figure 5.4 – Benchmarks. The two maps used to obtain topological graphs. Obstacles are in black. For each map, we generate a topological graph with a distance-based communication (range we used are depicted below the maps).

consequently after 10 agents, and CCBS stays above 40 %. However, on Office, SB is able to outperforms all algorithms after 10 agents. The performance of CCBS drops very fast on Office. DFS exhibit the same behaviors on both maps, not scaling beyond 10 agents.

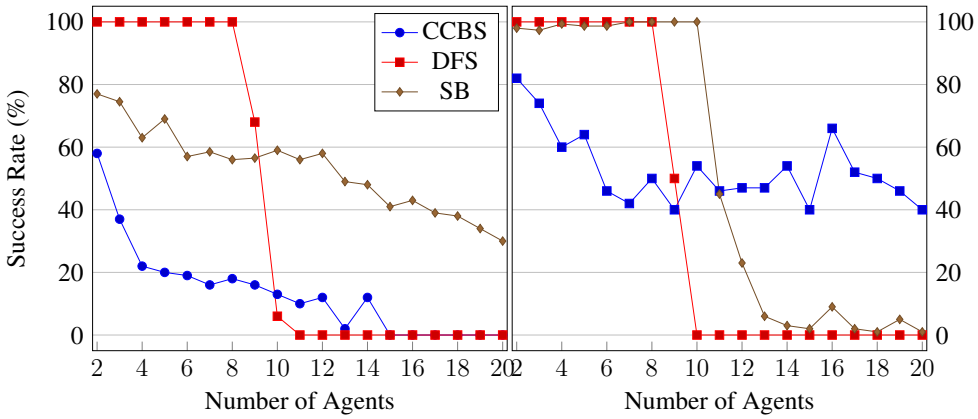
Now, we can observe that, as shown by Tateo et al. [151], SB generates very long executions, while DFS manages to generate shorter executions. However, we can see that DFS rapidly diverges from the optimal execution.

One would expect CCBS to be outperformed in all instances by approximate algorithms. However, quite surprisingly, CCBS is able to outperforms DFS and SB on Open with a relatively high number of agents. This may be the case as the decoupled structure of CCBS allow the generation of an almost-fully connected execution very early in the computation, while DFS must deal with state-explosion since the first step.

5.5 Conclusion

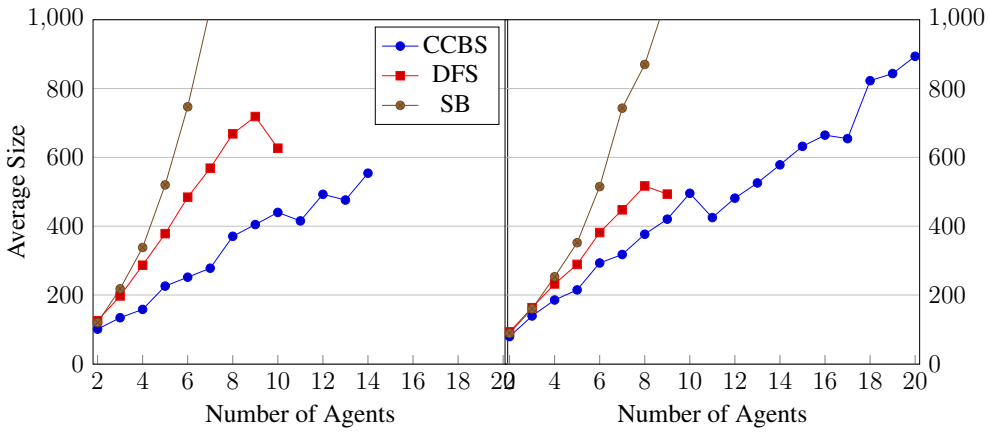
We presented CCBS, the first optimal algorithm for CMAPF which, in some cases, outperforms the state-of-the-art algorithms SB and DFS. The execution generated by CCBS are by an order of magnitude shorter than the most successful algorithm SB. On some maps, CCBS is able to scale to higher number of agents than other algorithms will still being optimal. Furthermore, the scalability displayed by CCBS may show that solvers of CMAPF benefit from a decoupled structure.

We believe that the scalability can be improved further by studied the known enhancement



(a) Offices Map (b) Open Map

Figure 5.5 – Success rate of CCBS, DFS, and SB to solve CMAPF on Offices and Open maps.



(a) Offices Map (b) Open Map

Figure 5.6 – Average execution size of DFS and CCBS to solve CMAPF on Offices and Open maps.

previously introduced for CBS. In CBS, a popular and efficient optimization considers *cardinal conflicts* [20], which are conflicts that can only be solved by increasing the lengths of the paths of all involved agents. This concept led to the integration of powerful heuristics in CBS [46, 90]. The adaptation of similar concepts for connectivity is however nontrivial. Furthermore, rather than only addressing connectivity conflicts, one could develop a combination of CCBS and CBS in order to solve CMAPF with collision constraints. Indeed, our approach makes it simple to interleave CCBS with CBS to solve connectivity and collisions during the same search. We leave this interesting research opportunity for future works. Our results open a challenging research avenue for developing such optimizations well adapted for both collision and connectivity constraints.

PART II

INCOMPLETE KNOWLEDGE

CHAPTER 6

CMAPF IN PARTIALLY-KNOWN ENVIRONMENTS

6.1 Introduction

The coordination of mobile agents is at the heart of many real world problems such as traffic control [39], robotics [43, 168], aviation [119] and more [142, 170]. Some of these problems have multiple aspects which make them complex: (1) Some systems are *multi-agent*, that is, the behaviors of agents influence others' and these influences must be taken into consideration when computing missions; this can be due, for instance, to collisions [101], sensor interferences [57, 135] etc.; (2) Some missions must ensure *connectivity*, that is, ensure periodic or constant connection to a station/agent to share acquired information [3]; (3) The environment may be only *partially known*, and the agents may discover it during the mission [127, 153]. Several works have considered problems containing these three aspects. For instance, several algorithmic approaches have been investigated to solve the coordination of multi-robot exploration [23, 161, 107]. Our objective in this contribution is to present a framework to study the theoretical complexity of planning problems with respect to these three aspects.

The theoretical complexity of some related problems have been studied in the literature. MAPF is an important framework introduced to study collision-free navigation of agents in warehouses [47, 101]. This problem was intensively studied and gave rise to a popular algorithm known as CBS [139]. An extension of MAPF with connectivity constraints, called CMAPF, was studied as well by Hollinger and Singh [69]. The complexity of CMAPF and algorithmic solutions were studied in Tateo et al. [151] and Charrier et al. [28]. However, CMAPF only addresses the multi-agent and connectivity aspects, and not the partial knowledge of the environment. The latter aspect is considered in the CTP, which is a well-known problem to study the navigation of an agent in a partially known graph [121]. While the initial framework was for a single agent, CTP has been extended to multiple agents in the settings of packet routing [74], multi-robot exploration [22] and more [98]. While a notion of communication was considered by Zhang and Xu [174] and Shiri and Salman [140], it is limited to settings where all agents can receive information at all times or only designated agents can send information. In contrast, we are interested in studying the setting where agents' ability to communicate depends on their positions in the graph, and in establishing theoretical complexity results of resulting problems.

In this chapter, we study the theoretical complexity of generating plans for a group of agents to reach a given target configuration. More precisely, we analyze the impact of enforcing or ignoring: (A) connectivity; (B) collision; (C) a bound on the length of the execution. For (A), we consider either *fully-connected* strategies, requiring that the agents remain connected at all times

	Decentralized	Connected
Bounded	NEXPTIME-complete (Th. 6.5)	PSPACE-complete (Th. 6.2, 6.3)
Unbounded	NEXPTIME-complete (Th. 6.4)	PSPACE-complete (Th. 6.1)

Table 6.1 – Complexity Results.

during the mission, or a *decentralized* strategy, allowing agents to disconnect and reconnect. (B) In some applications, collisions can be handled by a local collision avoidance system [69], and one can thus abstract away and ignore collisions in graph-based planning algorithms. (C) By providing a bound on the execution length, we can study the complexity of the decision problem associated to the optimization problem. Our results are summarized in Table 6.1. Interestingly, The PSPACE algorithm for the connected problem with a bounded execution is subtle and relies on a variant of the Savitch's theorem [134] we present here. Additionally, the PSPACE-completeness holds even in the case in which agents can always communicate, thus the hardness of the problem already comes from the incomplete knowledge of the movement graph. For the decentralized case, we prove the NEXPTIME-hardness in the bounded and unbounded cases by two separate reductions from the True Dependency Quantified Boolean Formula Problem (TDQBF) [122], thus showing that the problem becomes significantly harder in this case.

Let us compare our contribution with known complexity results. In the fully known environment, the CMAPF problem is PSPACE-complete in the connected and unbounded case [151], while it is NP-complete in the bounded case (with the bound given in unary) [69]. Thus, the partial knowledge of the environment does not render the problem harder in terms of complexity. In contrast, recall that in MAPF (without connectivity), one can check the existence of a solution in polynomial time [169], while the bounded problem is NP-hard [150], so the hardness is due to connectivity constraints. Some algorithms were presented for CMAPF by Tateo et al. [151] that can scale up to about ten agents. Since both problems are similar and belong to PSPACE, one can hope that these approaches can be extended to the partial knowledge case. On the other hand, our results show that the complexity of the decentralized case is significantly higher. While some tools and algorithms are available for Decentralized Partially Observable Markov Decision Processes (Dec-POMDPs), which are in the same complexity class, the scalability is limited and the development of efficient algorithms for this case seems more challenging (see survey of Amato et al. [2]).

6.2 Our framework

6.2.1 Modeling Incomplete Knowledge

To formalize CMAPF in the incomplete knowledge setting, let us show how to represent the initial knowledge of the agents, and how their knowledge evolves during the execution. Agents initially know the exact set of vertices, but only have a lower and an upper approximation of the

actual graph: they know that some (communication or movement) edges are *certain* (they must be present), while some are *uncertain* (they may be absent).

Definition 6.1 (Initial Knowledge)**.** *The initial knowledge is modeled by a pair of topological graphs (G_1, G_2), with the graph $G_1 = \langle V, E_{m1}, E_{c1}\rangle$ a lower bound, and $G_2 = \langle V, E_{m2}, E_{c2}\rangle$ an upper bound on the knowledge about the actual graph with $E_{m1} \subseteq E_{m2}$ and $E_{c1} \subseteq E_{c2}$.*

The agents initially know G_1 and G_2 while the actual graph $G = \langle V, E_m, E_c\rangle$ is initially unknown to them. They only know that $E_{m1} \subseteq E_m \subseteq E_{m2}$ and $E_{c1} \subseteq E_c \subseteq E_{c2}$, written as $G_1 \subseteq G \subseteq G_2$. The perfect information case is captured by $G_1 = G_2 (= G)$.

A movement (resp. communication) edge is said to be *certain* (i.e. sure to be present) if it is in E_{m1} (resp. E_{c1}); it is said *uncertain* (i.e. can be absent) if it is in $E_{m2} \setminus E_{m1}$ (resp. $E_{c2} \setminus E_{c1}$). We assume that the communication edges of the actual graph are undirected, so for all $(u, v) \in E_{c2} \setminus E_{c1}$, either $(u, v), (v, u) \in E_c$, or $(u, v), (v, u) \notin E_c$.

We say that an edge (u, v) is an *uncertain undirected movement edge* when $(u, v), (v, u) \in E_{m2} \setminus E_{m1}$. The environment can leave both edges $(u, v), (v, u)$, remove both edges $(u, v), (v, u)$, but also remove (u, v) and leave (v, u), or remove (v, u) and leave (u, v). That is, the movements edges of the actual graph are not necessarily undirected.

Example 6.1. *Figure 6.1a depicts an initial knowledge (G_1, G_2). The area is divided in two zones connected by two bridges, represented by the edges (s_2, s_4) and (s_3, s_5), with an uncertainty on their traversability and on the communication between s_1 and s_4.*

A strategy σ_a for agent a tells where to go next after a given execution e. Formally:

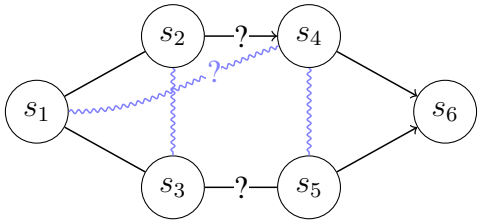
Definition 6.2 (Strategy)**.** *A* strategy *σ_a for agent a maps any execution e to a vertex such that $(v, \sigma_a(e)) \in E_m$ where v is the vertex at which agent a is in the last configuration of e.*

A *joint strategy* π is a tuple $\langle \pi_1, \ldots, \pi_n\rangle$ where π_i is a strategy for agent i. The *outcome* of a joint strategy π starting from configuration c^s is the execution e defined by induction as follows: $e[0]$ is c^s, and for $t \geq 1$, $e[t]$ is the configuration in which agent i is at vertex $\pi_i(e[0..t-1])$.

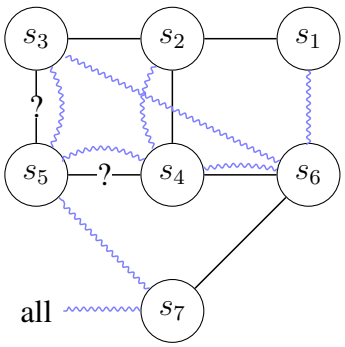
In the context of incomplete knowledge, the behaviors of the agents only depend on their observations, similarly to imperfect information games as done by Berthon et al. [16]. The strategies, as defined above, do not necessarily take observations of the agents into account. We will now formalize observations and *uniform* strategies, that is, those respecting the observations of the agents.

In our setting, at any time, an agent observes all movement edges adjacent (both in- and out-coming) to the vertex v it occupies. Moreover, they observe the presence or absence of a communication edge between v and v' if v' is occupied by another agent with which there is a direct or indirect communication (via other agents). Intuitively, during an execution, at each step, each agent updates their knowledge about the graph with these observations they receive. Moreover, they share all their knowledge with all agents with which they are connected at each step.

The observation of adjacent movement edges has been a recurrent practice in theoretical works [121, 15] as well as robotics [40, 82], and our formalism is inspired from these works.



(a) An initial knowledge (G_1, G_2).



(b) An initial knowledge of a topological graph. Vertex s_7 has communication edges with all other vertices.

⟶ Certain movement edge
— Certain undirected movement edge
-?→ Uncertain movement edge
-?- Uncertain undirected movement edge
〰 Certain communication edge
〰?〰 Uncertain communication edge

Figure 6.1 – Examples of partially known topological graphs.

The knowledge of an agent at any time corresponds intuitively to a pair of graphs as in Definition 6.1. For agent a and execution e, let us denote by $k_a(e)^G$ the knowledge of agent a about the graph after observing the execution e in actual graph G. Given such knowledge $K = k_a(e)^G$, the agent can deduce an under-approximation and an over-approximation of the actual graph; let us denote these by $\underline{G}^K$ and $\overline{G}^K$ respectively. In particular, if K is the knowledge the agents have initially, then $\underline{G}^K = G_1$ and $\overline{G}^K = G_2$. We present a representation of the knowledge using predicates in the Subsection 6.2.2 where a detailed formalization of $k_a(e)^G$ can be found.

In the rest of the chapter, we assume all considered strategies to be *uniform*, that is, they comply with the knowledge of the agents: the strategies prescribe the same move to all executions that are indistinguishable with the agent's observations. Formally, if $|e| = |e'|$ and $k_a(e)^G = k_a(e')^G$, then $\sigma_i(e) = \sigma_i(e')$.

6.2.2 Complements

We present here a formal modeling of the incomplete knowledge in our setting.

In our setting, at any time, an agent observes all movement edges adjacent (in- and outcoming edges) to the vertex v it occupies. Moreover, they observe the presence or absence of a communication edge between (v, v') if v' is occupied by another agent with which there is a direct or indirect communication (via other agents). Intuitively, during an execution, at each step, each agent updates their knowledge about the graph with these observations they receive.

Moreover, they share all their knowledge with all agents with which they are connected at each step.

Given graph $G = \langle V, E_m, E_c \rangle$, let us define the direct observation $obs_a(c)$ of agent a at a configuration c to be the set:

$$\left\{o^m_{c_a,v'} \mid (c_a, v') \in E_m\right\} \cup \left\{o^{\neg m}_{c_a,v'} \mid (c_a, v') \notin E_m\right\} \quad (1)$$

$$\cup \left\{o^m_{v',c_a} \mid (v', c_a) \in E_m\right\} \cup \left\{o^{\neg m}_{v',c_a} \mid (v', c_a) \notin E_m\right\} \quad (2)$$

$$\cup \left\{o^c_{c_a,c_b} \mid j \text{ is an agent and } (c_a, c_b) \in E_c\right\} \quad (3)$$

$$\cup \left\{o^{\neg c}_{c_a,c_b} \mid j \text{ is an agent connected to } i \text{ in } c, (c_a, c_b) \notin E_c\right\} \quad (4)$$

where $o^m_{u,v}$, $o^{\neg m}_{u,v}$, $o^c_{u,v}$ and $o^{\neg c}_{u,v}$ are abstract terms that represent the *observations*. In the definition of $obs_a(c)$, points (1) and (2) mean that agent a directly observes the set of movement edges adjacent to her current position. Point (3) means that agent a observes a communication edge when she can communicate with another agent b. Point (4) means that agent a observes the absence of a communication edge when she sees that she can not communicate directly with another agent b but can communicate with b via multi-hop. That is, we say that j is an agent connected to i in c when there is a communication path $c_{i_1}, c_{i_2}, \ldots, c_{i_k}$ with $i_1 = i$ and $i_k = j$.

Let O denote the set of all observations. We define the knowledge of an agent as a subset of O. We define the *initial knowledge for the pair* (G_1, G_2) as follows:

$$K^0(G_1, G_2) = \left\{o^m_{u,v} \mid (u, v) \in E_{m1}\right\} \cup \left\{o^c_{u,v} \mid (u, v) \in E_{c1}\right\} \cup \left\{o^{\neg m}_{u,v} \mid (u, v) \notin E_{m2}\right\} \cup \left\{o^{\neg c}_{u,v} \mid (u, v) \notin E_{c2}\right\}$$
$$\text{where } G_i = \langle V, E_{mi}, E_{ci} \rangle.$$

This corresponds to the *a priori* knowledge on the graph all agents have before making any observation.

During the execution, agents update their knowledge at each step, upon visiting a new configuration. Formally, the knowledge $k_a((K_j)_{1 \le j \le n}, e)^G$ of agent a after observing execution e in actual graph G, where each agent j starts with initial knowledge K_j, is defined by induction on e:

— $k_a((K_j)_{1 \le j \le n}, \epsilon)^G = K_a$

— $k_a((K_j)_{1 \le j \le n}, ec)^G$ is the union of:

$$k_a((K_j)_{1 \le j \le n}, e)^G; \quad (a)$$

$$obs_a(c); \quad (b)$$

$$\bigcup_{j \text{ connected to } i \text{ in } c} (k_b((K_j)_{1 \le j \le n}, e)^G \cup obs_b(c). \quad (c)$$

Intuitively, the knowledge of an agent is composed of (a) her knowledge collected until now, (b) her current observation, and (c) the knowledge of the agents she is connected to and their current observation.

When the agents start with an initial knowledge (G_1, G_2) that is clear from the context, we will omit the tuple $(K_j)_{1\leq j\leq n}$ and simply write $k_a(\pi)^G$.

Note that during a connected execution, all agents have an identical knowledge at all times. We thus omit the subscript i, and replace the tuple of initial knowledge sets by a single set K and write $k(K, \pi)^G$ rather than $k_a((K_j)_{1\leq j\leq n}, \pi)^G$.

6.2.3 Decision Problems

We consider the decision problem of reaching a configuration c^g from a configuration c^s in less than k steps, using uniform strategies. For two configurations c^s, c^g, let us call a topological graph G (c^s, c^g)-*admissible* if there is an execution from c^s to c^g, which is not necessarily connected.

Definition 6.3 (Positive Instance)**.** *We say that an instance (G_1, G_2, c^s, c^g, k) is positive if there exists a joint strategy π such that in all (c^s, c^g)-admissible graphs G satisfying $G_1 \subseteq G \subseteq G_2$, the outcome of π starting in c^s ends in c^g in less than k steps.*

Observe that the above problem requires that a strategy ensures the reachability of the target configuration only for graphs that are (c^s, c^g)-admissible and compatible with the initial knowledge. In fact, intuitively, we would like the strategy to work under all possible graphs G with $G_1 \subseteq G \subseteq G_2$. However, requiring a strategy to ensure reachability in a non-admissible graph does not make sense, since even a strategy with full information would fail. We thus require the strategies to make their best efforts, that is, to ensure the objective unless it is physically impossible.

Example 6.2. *Consider the example of Figure 6.1a. If both bridges (i.e. movement edges (s_2, s_4) and (s_3, s_5)) are absent in the actual graph, the graph is not (s_1, s_6)-admissible and there cannot be a strategy ensuring reachability. The admissible graphs contain either (s_2, s_4), or (s_3, s_5), or possibly both. Note that, this instance is negative for $k < 6$ (that is, it does not admit a solution). Indeed, consider a strategy that moves the agent, for instance, to s_2. In the graph where (s_2, s_4) is absent, the agent would need to come back to s_1 and take the alternative path, which requires an execution of total length 6; and the situation is symmetric if the first move is towards s_3. The instance is nonetheless positive for $k \geq 6$ with the described strategy. However, if the edges (s_2, s_1) and (s_3, s_1) were not present, then the instance would be negative. In fact, once the agent moves to s_2 or s_3, they get stuck if the graph only contains the other bridge.*

We now define the *connected* version of Definition 6.3. For two configurations c^s, c^g, we say that a topological graph G is (c^s, c^g)-*c-admissible* if there is a connected execution from c^s to c^g. We will often omit the pair of configurations which will be clear from the context, and write simply admissible or c-admissible.

Definition 6.4 (c-Positive Instance)**.** *We say that an instance (G_1, G_2, c^s, c^g, k) is c-positive if there exists a joint strategy π such that in all (c^s, c^g)-c-admissible graphs G satisfying $G_1 \subseteq G \subseteq G_2$, the outcome of π starting in c^s is connected and ends in c^g in less than k steps.*

In the connected case, agents cannot visit a disconnected configuration. Hence, the considered strategies only visit configurations that are certainly connected. Observe that agents can make observations about the presence or absence of communication edges while being connected and use this information later.

Example 6.3. *Let us illustrate the above property on the example of Figure 6.1b. Assume there are two agents, the starting and goal configurations are $c^s = \langle s_1, s_6\rangle$ and $c^g = \langle s_5, s_7\rangle$, and the only uncertainty is about the movement edges (s_3, s_5) and (s_4, s_5). Here, Agent 2 could immediately move to her target s_7; however, she could also cooperate with Agent 1 and lower the total completion time. Indeed, from their start configuration $\langle s_1, s_6\rangle$, the agents first move to $\langle s_2, s_4\rangle$ where Agent 2 observes whether (s_4, s_5) is present. Assume the edge is present. Then, they follow the sequence $\langle s_4, s_6\rangle \cdot \langle s_5, s_7\rangle$; and otherwise $\langle s_3, s_6\rangle \cdot \langle s_5, s_7\rangle$. Thus, in order to minimize the length of the execution, the agents do not always take their shortest paths but might help other agents by obtaining information about the graph.*

Consider now the same example in which the communication edge (s_3, s_6) is uncertain. If this edge is absent then Agent 2 cannot help Agent 1 achieve the target faster since if the former moves to s_4 and (s_4, s_5) is absent, then, in order to maintain connectivity, the next configurations should be $\langle s_4, s_6\rangle \cdot \langle s_2, s_7\rangle \cdot \langle s_3, s_7\rangle \cdot \langle s_5, s_7\rangle$. An execution of the same size is obtained when Agent 2 moves to s_7 in the first step.

For both Definitions 6.3 and 6.4 above, let us call a joint strategy a *witness* if it witnesses the fact that the given instance is positive, and respectively, c-positive.

We instantiate the CMAPF problem in four different settings. The four following decision problems are defined depending on whether we consider the connectivity requirement and whether the bound is finite. Note that the bounded problems are the decision problems associated to the optimization problems.

Problem 6.1 (bDMAPFI).

- Input: *A pair (G_1, G_2), a start configuration c^s, a goal configuration c^g and a bound $k < \infty$.*
- Output: *Is (G_1, G_2, c^s, c^g, k) positive?*

Problem 6.2 (bCMAPFI).

- Input: *A pair (G_1, G_2), a start configuration c^s, a goal configuration c^g and a bound $k < \infty$.*
- Output: *Is (G_1, G_2, c^s, c^g, k) c-positive?*

Problem 6.3 (DMAPFI).

- Input: *A pair (G_1, G_2), a start configuration c^s and a goal configuration c^g.*
- Output: *Is $(G_1, G_2, c^s, c^g, \infty)$ positive?*

Problem 6.4 (CMAPFI).

- Input: *A pair (G_1, G_2), a start configuration c^s and a goal configuration c^g.*
- Output: *Is $(G_1, G_2, c^s, c^g, \infty)$ c-positive?*

As we will see, the encoding of the integer k does not change the overall complexity. Lower bounds are all obtained directly for the unary encoding; the lower bounds for the binary encoding follows. Concerning the upper bounds, we explain for each case how to design an algorithm with k encoded in binary.

6.3 Quantified Boolean Formula

In this section, we present the True Quantified Boolean Formula Problem (TQBF) and True Dependency Quantified Boolean Formula Problem (TDQBF). These problems are necessary for the reductions presented in the following sections.

6.3.1 Quantified Boolean Formula

The True Quantified Boolean Formula Problem (TQBF) asks whether a given a QBF φ is valid. Recall that a QBF φ is of the form $\forall z_1 \exists z_2 \dots Q_n z_n \psi$ where $z_1, \dots, z_n$ are Boolean variables; Q_n is $\forall$ if n is odd, and $\exists$ if n is even; ψ is a Boolean formula in Conjunctive Normal Form (CNF) over variables $z_1, \dots, z_n$. TQBF is PSPACE-complete [147]. For convenience and simplicity, we say that an existentially (resp. universally) quantified variable is an existential (resp. universal) variable. We will often use $x_1, x_2, \dots$ for existential variables and $y_1, y_2, \dots$ for universal ones.

6.3.2 Dependency Quantified Boolean Formula

The TDQBF is the problem of deciding whether a given DQBF is valid. A DQBF is a formula in which dependencies of existential variables over universal variables are explicitly specified. A DQBF is of the form $\forall y_1, \dots, y_n \exists x_1(O_{x_1}) \dots \exists x_n(O_{x_n})\ \psi$, where each O_{x_i} is, the *dependency set*, a subset of universally quantified variables, and ψ is a Boolean formula in CNF over $x_1, \dots, x_n, y_1, \dots, y_n$. It is worth noting that a QBF can be seen as a DQBF with $O_{x_1} \subseteq O_{x_2} \subseteq ... \subseteq O_{x_n}$.

Formally, a DQBF φ is valid iff there exists a collection of Skolem functions $A = (A_{x_i} : \{0,1\}^{O_{x_i}} \to \{0,1\})_{i=1..n}$ such that replacing each existential variable x_i by a Boolean formula representing A_{x_i}, turns ψ into a tautology. TDQBF is NEXPTIME-complete [122], and will be used to prove NEXPTIME lower bounds in Section 6.5.

6.4 Connected Reachability

We first address the case where agents must be connected at each step of the execution. In this case, agents share their knowledge at all times and thus the group of agents can be considered as a single agent playing against the environment.

6.4.1 Unbounded Case

We first focus on the existence of an unbounded connected strategy. Interestingly, we show that verifying the existence of a connected strategy in a partially known environment is not harder than in a perfectly known environment.

Theorem 6.1. ***CMAPFI** is PSPACE-complete.*

We define the following recursive property: $P(\underline{G}, \overline{G}, c, c^g)$ holds for graphs $\underline{G}, \overline{G}$, and configurations c, c^g if either $\overline{G}$ is not (c^s, c^g)-c-admissible, or there exists a connected execution π from c to c^g in $\overline{G}$ such that for all G with $\underline{G} \subseteq G \subseteq \overline{G}$, writing K_0 for the initial knowledge for the pair $(\underline{G}, \overline{G})$, there exists $0 \leq i_0 \leq |\pi|$ with $k(K_0, c)^G = k(K_0, \pi[0..i_0 - 1])^G$, and either $\pi[i_0] = c^g$ or $(k(K_0, c)^G \subsetneq K = k(K_0, \pi[0..i_0])^G$ and $P(\underline{G}^K, \overline{G}^K, \pi[i_0], c^g))$.

The following two lemmas prove Theorem 6.1.

Lemma 6.1. *An instance $I = (G_1, G_2, c^s, c^g, \infty)$ is c-positive if, and only if $P(G_1, G_2, c^s, c^g)$.*

Proof. We prove the following more general property by induction on the number of edges present in G_2 and absent in G_1, that is, $|E_{m2}| - |E_{m1}| + |E_{c2}| - |E_{c1}|$: For all graphs $G_1 \subseteq \underline{G} \subseteq \overline{G} \subseteq G_2$, and configurations c, if the instance $(\underline{G}, \overline{G}, c, c^g, \infty)$ is c-positive then $P(\underline{G}, \overline{G}, c, c^g)$.

If $\underline{G} = \overline{G}$ then either the instance is not c-admissible and, then, $P(\underline{G}, \overline{G}, c, c^g)$ holds, or there is a connected execution in $\overline{G}$ from c to c^g in which case the property holds as well.

Assume $\underline{G} \subsetneq \overline{G}$, and consider σ a witness strategy for the instance $(\underline{G}, \overline{G}, c, c^g, \infty)$. Consider a graph $\underline{G} \subseteq G \subseteq \overline{G}$. If the execution π induced by σ does not reveal any new observation in G, then it must end in c^g and $P(\underline{G}, \overline{G}, c, c^g)$ holds. Otherwise, let i_0 be the first step where a new observation is made. Since σ is a witness strategy, it is a witness for the instance $(\underline{G}^K, \overline{G}^K, \pi[i_0], c^g, \infty)$ as well where $K = k(\pi[0..i_0])^G$. Since $\overline{G}^K$ and $\underline{G}^K$ have a smaller number of differences than $\overline{G}$ and $\underline{G}$, we conclude by induction that $P(\underline{G}, \overline{G}, \pi[i_0], c^g)$. Thus, $P(\underline{G}, \overline{G}, c, c^g)$ holds.

Let us now show that all instances that satisfy the property are c-positive. Assume that the property $P(G_1, G_2, c^s, c^g)$ holds. We define the joint strategy σ on all graphs $\overline{G}^K$ and executions π in $\overline{G}^K$, such that $P(\underline{G}^K, \overline{G}^K, \mathsf{last}(\pi), c^g)$, by induction on the length of π.

Assume σ is constructed for an execution π and knowledge K. If $\overline{G}^K$ is not c-admissible, then any strategy is a witness strategy so σ can be defined arbitrarily. Otherwise, consider the connected execution π' given by $P(\underline{G}^K, \overline{G}^K, \mathsf{last}(\pi), c^g)$. We define σ so that agents follow π' until index i_0, in which case either $\pi'[i_0] = c^g$ or $P(\underline{G}^{K'}, \overline{G}^{K'}, \pi\pi'[1..i_0], c^g)$. □

Lemma 6.2. *$P(G_1, G_2, c^s, c^g)$ can be checked in polynomial space.*

Proof. The existence of a connected execution can be checked in polynomial space by Theorem 3.2. However, the size of such an execution can be exponential, and checking the property $P(G_1, G_2, c^s, c^g)$ requires iterating over the step of the execution. We thus need to combine the enumeration of the connected execution as we check P recursively.

The procedure to check $P(\underline{G}, \overline{G}^K, c, c^g)$ works as follows. We non-deterministically guess a connected execution step by step, from c to c^g using the PSPACE algorithm of Theorem 3.2. We thus only keep the last configuration in memory, a binary integer counter to bound the length of the execution (bounded by the number of configurations, thus an exponential), and the current graph $\overline{G}^K$. If there is no such execution, we accept. Otherwise, at each step, say, after having visited execution π and generated next configuration c' we enumerate all possible sets $K' = k(K_0, \pi c')^G$, where K_0 is the initial knowledge for the pair (G_1, G_2). This can be done by enumerating all subsets of movement edges adjacent to c', present in $\overline{G}^K$ but not

in $\underline{G}^K$, and similarly communication edges revealed by c'. Note that $k(K_0, \pi c')^G$ only depends on $k(K_0, \pi)^G$ and c' which the algorithm already has. There is an exponential number of possibilities, and these can be enumerated in polynomial space. For each case, we check recursively whether $P(\underline{G}^{K'}, \overline{G}^{K'}, c', c^g)$. Since the knowledge can increase only a polynomial number of times (since the knowledge can only increase when an edge is added or removed), the depth of the recursive calls is polynomial. Thus, overall, the procedure uses polynomial space. □

6.4.2 Bounded Case

We now study the existence of a bounded connected strategy. We show that this problem is PSPACE-complete even when the communication graph is complete.

Theorem 6.2. ***bCMAPFI*** *is in PSPACE when the bound is given in binary.*

Proof. Let us first prove the upper bound when k is given in unary. As APTIME = PSPACE [27], we give an alternating algorithm that runs in polynomial time, as follows. At each step, the existential player chooses the next connected configuration to move the agents; and the universal player chooses the information about the newly discovered edges. After k steps the algorithm accepts if the target configuration is reached, or the revealed edges mean that the graph is not c-admissible. The number of steps is bounded by k, which is polynomial, thus the algorithm runs in polynomial time.

There is one subtlety to prove the correctness. The alternating algorithm actually corresponds to a slight variant of our setting which can be seen as a game. In our setting, the environment chooses a graph G with $G_1 \subseteq G \subseteq G_2$ at the beginning, and the agents discover the graph G as they move. In contrast, in the alternating algorithm, the universal player reveals the graph step by step; therefore the environment might adapt the graph to the moves of the existential player.

Lemma 6.3. *The alternating algorithm is correct.*

Proof. First, observe that if the existential player has a strategy σ in the alternating algorithm, then the instance is c-positive. In fact, for any graph G with $G_1 \subseteq G \subseteq G_2$, consider strategy τ of the universal player which makes choices according to G. Since this σ wins against τ, either the graph is not admissible or the agents successfully arrive to the target configuration. Conversely, assume that the instance is c-positive, that is, for all choices of an admissible graph G, the agents arrive at a target configuration under some joint strategy σ. We apply σ in the alternating algorithm. Consider any strategy τ of the universal player, and observe the execution induced by (σ, τ). If the graph induced by τ is revealed not to be admissible, then the existential player wins. Otherwise, consider any admissible graph G with $G_1 \subseteq G \subseteq G_2$ compatible with the edges revealed during the execution of (σ, τ). Since σ is winning in the original game when the underlying graph is G, the existential player also wins. △

When k is binary, the previous algorithm does not run in polynomial time. However, observe that the number of alternations can be bounded by a polynomial because there is only a polynomial number of steps in which the universal players reveal *new* information to the coalition

of agents. In fact, the universal player is only useful when some agent is at a vertex that has not been seen before, and this can only happen a linear number of times. Furthermore, the previous algorithm runs in polynomial space.

When k is binary, our problem is in $\mathsf{STA}(\mathsf{poly}(n), *, \mathsf{poly}(n))$ where $\mathsf{STA}(s(n), t(n), a(n))$ is the set of problems decided in space $O(s(n))$, time $O(t(n))$ with $O(a(n))$ alternations. Our problem is in PSPACE thanks to the generalization of Savitch's theorem we prove:

Lemma 6.4. $\mathsf{STA}(\mathsf{poly}(n), *, \mathsf{poly}(n)) \subseteq \mathsf{PSPACE}$.

Proof. To build a PSPACE algorithm, we perform a DFS of the computation tree T in a succinct manner. Consider the tree T', built from T, where we only keep vertices in which the universal player makes a decision, while paths along which only the existential player moves are shortcut into single edges. The depth of this tree is polynomial by definition.

The idea is to run a DFS on T' to check whether the machine accepts. This can be done in polynomial space provided that the children of all vertices can be computed in polynomial space. Successors of an existential configuration c in T' are computed as follows: we generate on-the-fly all possible configurations c' and test whether c' is reachable from c in the original alternating machine by using a PSPACE oracle. The DFS that runs in PSPACE augmented with this PSPACE oracle gives a polynomial space procedure. △

Intuitively, this lemma is proved by guessing the computations between each universal choice by a PSPACE oracle, which yields an overall PSPACE algorithm.

□

Theorem 6.3. ***bCMAPFI*** *with complete connectivity is PSPACE-hard.*

Proof. The lower bound is proven by reduction from TQBF.

Consider a QBF φ of the form $\forall z_1 \exists z_2 \dots Q_n z_n \psi$ where ψ is a Boolean formula in conjunctive normal form with n variables and m clauses.

In the reduction, we call a *movement path*, from node v to node u, a chain of nodes linking v to u by movement edges. In addition, we denote an occurrence of a positive (resp. negative) literal of a variable z, by $\underline{z}$ (resp. $\underline{\neg z}$)

We create the graphs G_1 and G_2 as described in Figure 6.2. More precisely, for each variable z, we create a gadget shown in Figure 6.2a. For each clause, we create a gadget as depicted in Figure 6.2b. In addition, we create a movement edge between a node $v_{\underline{z}}$ (resp. $v_{\underline{\neg z}}$) and a node t_γ if the literal z (resp. $\neg z$) is present in the clause γ.

We define the initial and target configurations c^s, c^g as follows. There is a single agent at each vertex of the form s_γ (resp. s_z) whose target is t_γ (resp. t_z). Furthermore, at each $s_{\underline{z}}$ (resp. $s_{\underline{\neg z}}$) there is one agent for each clause that contains z (resp. $\neg z$) and her target is $t_{\underline{z}}$ (resp. $t_{\underline{\neg z}}$).

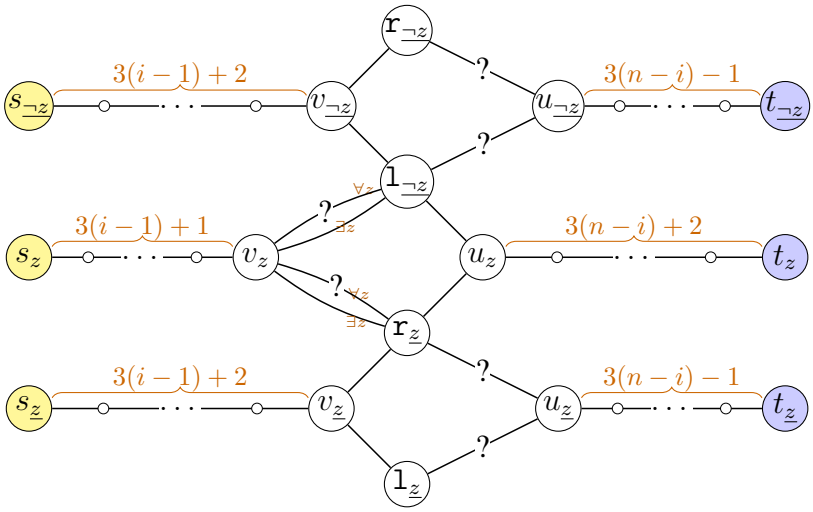
We show that φ is true iff (G_1, G_2, c^s, c^g, k) is c-positive with $k = 3(n-1) + 5$.

For simplicity, we classify the agents as follows.

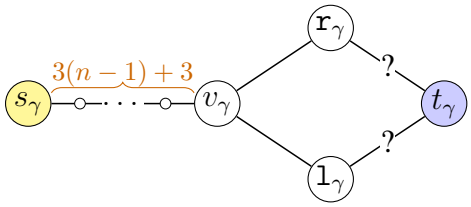
— A *Variable agent* is an agent starting at a node s_z;

— a *Clause agent* is an agent starting at a node s_γ;

— a *Positive (resp. negative) occurrence agent* is an agent starting at $s_{\underline{z}}$ (resp. $s_{\underline{\neg z}}$).



(a) Variable gadget for variable z. Edges between v_z and $\mathtt{l}_{\neg\underline{z}}$, and between v_z and $\mathtt{r}_{\underline{z}}$ are both certain if z is existential and both uncertain if z is universal.



(b) Clause gadget γ.

—— Certain Movement edge

–?– Uncertain Movement edge

$\overset{\forall_z}{<?>}_{\exists_z}$ Movement edge that is certain if z is existential and uncertain if z is universal

$(v_{\underline{z}}, t_\gamma) \in E_{m1}$ when $\underline{z}$ is an occurrence in γ
$(v_{\neg\underline{z}}, t_\gamma) \in E_{m1}$ when $\neg\underline{z}$ is an occurrence in γ

Figure 6.2 – Gadgets for the reduction from TQBF into the bounded reachability problem in the complete connectivity case.

($\Rightarrow$) Assume that the QBF φ is true. There exists a collection of Skolem functions A such that for each existential variable z_i (where i is even), and an assignment ν to universally quantified variables in $z_1, z_3, \ldots, z_{i-1}$, $A_{z_i}(\nu) \in \{\top, \bot\}$ is the value assigned to z_i such that φ is true under assignment ν augmented with the values of A. We construct the following strategy σ, which guarantees that c^g is reached in k steps from c^s.

Intuitively, the lengths of the initial movement paths are designed so that agents starting at s_{z_i} arrive at v_{z_i} in the order of their indices. For an existential variable agent, the choice of the successor from v_{z_i} determines the value of z_i; while for a universal variable agent, the choice is made by the environment. More precisely, if the agent moves to $\mathtt{r}_{\underline{z_i}}$, then z_i is set to true, if she moves to $\mathtt{l}_{\underline{\neg z_i}}$ it is set to false.

Formally, all agents start by moving to their respective v nodes (e.g. A clause agent at s_γ moves to v_γ). They arrive at these nodes at different moments due to the sizes of their movement paths. An existential variable agent arrives at node v_{z_i} at time $3(i-1)+1$. At this point, all universal variable agents among $z_1, z_3, \ldots, z_{i-1}$ have arrived to their respective nodes v_{z_j}, thus revealing the values of these variables. Let ν be this assignment. If $A_{z_i}(\nu) = \top$, the agent moves to $\mathtt{r}_{\underline{z_i}}$, and otherwise, to $\mathtt{l}_{\underline{\neg z_i}}$.

When a universal variable agent arrives to v_{z_i} at time $3(i-1)+1$, she follows the only edge dictated by the environment, either to $\mathtt{r}_{\underline{z_i}}$ or to $\mathtt{l}_{\underline{\neg z_i}}$. This assigns $\top$ to the variable z_i in the former case, and $\bot$ in the latter case. Observe that if the graph is admissible, then there must exist a path from source to target for each agent; this means that one of these edges must be present.

Consider a positive (resp. negative) occurrence agent associated to a clause γ. This agent arrives to $v_{\underline{z_i}}$ (resp. $v_{\underline{\neg z_i}}$) at time $3(i-1)+2$. Observe that the variable agent has already determined the value of z_i in the previous step.

• if z_i is assigned $\top$ (resp. $\bot$) then the agent moves to t_γ, observe which edges are available at t_γ, and immediately comes back to $v_{\underline{z_i}}$ (resp. $v_{\underline{\neg z_i}}$). Now, the edge between $\mathtt{r}_{\underline{z_i}}$ (resp. $\mathtt{l}_{\underline{\neg z_i}}$) and $u_{\underline{z_i}}$ (resp. $u_{\underline{\neg z_i}}$) has been observed by agent z_i; so if this edge is present, she moves to $\mathtt{r}_{\underline{z_i}}$ and $u_{\underline{z_i}}$; and if not, then the edge from $\mathtt{l}_{\underline{z_i}}$ to $u_{\underline{z_i}}$ must be available, and she arrives to $t_{\underline{z_i}}$ (resp. $t_{\underline{\neg z_i}}$) at time k.

• if z_i is assigned $\bot$ (resp. $\top$) then she does not visit t_γ, but moves to $\mathtt{l}_{\underline{z_i}}$ (resp. $\mathtt{l}_{\underline{\neg z_i}}$). If the edge between $\mathtt{l}_{\underline{z_i}}$ (resp. $\mathtt{l}_{\underline{\neg z_i}}$) and $u_{\underline{z_i}}$ (resp. $u_{\underline{\neg z_i}}$) is available, she moves to $t_{\underline{z_i}}$ (resp. $t_{\underline{\neg z_i}}$), otherwise she moves back and reach $t_{\underline{z_i}}$ (resp. $t_{\underline{\neg z_i}}$) at time k, through $\mathtt{r}_{\underline{z_i}}$ (resp. $\mathtt{r}_{\underline{\neg z_i}}$).

It remains to argue that clause agents can reach their target nodes within k steps. Since φ is true, by the definition of A, whatever the choice for the universal variables, some literal ℓ of each clause γ is assigned to true. Therefore, the positive or negative occurrence agent corresponding to this literal visits t_γ, thus revealing the edges available from t_γ to $\mathtt{r}_\gamma$ and $\mathtt{l}_\gamma$. Note that at least one of these edges must be available for the graph to be admissible. Thus, a clause agent arriving to v_γ at time $3(n-1)+3$ can follow the available path to reach t_γ exactly at time k.

($\Leftarrow$) Let σ be a witness joint strategy. Following σ, each clause agent c must know the available edges in the rest of their paths at time $3(n-1)+3$ since otherwise they cannot ensure reaching t_γ at time k. Thus, for each clause γ, the node t_γ is visited by some occurrence agent under strategy σ. Furthermore, an occurrence agent $\underline{z}$ (resp. $\underline{\neg z}$) can visit node t_γ and still make it to $t_{\underline{z}}$ (resp. $t_{\underline{\neg z}}$) in time iff the associated variable agent has observed the presence of the edge

between $u_{\underline{z}}$ (resp. $u_{\underline{\neg z}}$) and $\mathtt{r}_{\underline{z}}$ (resp. $\mathtt{l}_{\underline{\neg z}}$) beforehand. In fact, otherwise, if the occurrence agent makes a wrong guess between $\mathtt{l}_{\underline{z}}$ and $\mathtt{r}_{\underline{z}}$, they will not arrive to $t_{\underline{z}}$ (resp. $t_{\underline{\neg z}}$) at time k. Hence, the joint strategy of the variable agents determines an assignment function which satisfies φ.

□

Our reduction actually builds an undirected movement graph. Thus, PSPACE-hardness holds already for undirected movement graphs. Note that in our current setting, pairs of uncertain edges of the form (u, v) and (v, u) are treated separately, but the lower bound proof still holds when they are seen as one.

6.5 Decentralized Reachability

We now tackle the case where agents are allowed to be disconnected; at each configuration, they share their knowledge with all agents to which they are connected. This case is harder because agents no longer follow a centralized strategy and they must cooperate to exchange information at the right moment to reach their targets.

6.5.1 Unbounded Case

Theorem 6.4. ***DMAPFI*** *is NEXPTIME-complete.*

Proof. For the upper bound, an NEXPTIME algorithm consists in guessing uniform strategies for all agents and checking whether the joint strategy is a witness. Such a strategy has exponential size since it is a function of the sets of knowledge of the agents and the current vertex. One can enumerate all graphs G between G_1 and G_2, and execute the joint strategy on G to check that it ensures the reachability of the target. Moreover, the executions to be checked have at most exponential length. In fact, executions can be seen as paths in a meta-graph where vertices are configurations augmented with the sets of knowledge of the agents. This meta-graph is of exponential size, so it is sufficient to consider executions of exponential length. The overall non-deterministic algorithm is thus in exponential time.

The lower bound is shown by reduction from TDQBF. Given a DQBF φ of the form $\forall y_1, \dots, y_n \exists x_1(O_{x_1}) \dots \exists x_n(O_{x_n})\ \psi$, we build an instance (G_1, G_2, c^s, c^g, k) of unbounded decentralized reachability. We denote by $\gamma_1, \dots, \gamma_m$ the clauses in ψ.

The graph G_1 and G_2 as follows. For each variable z, we create the gadget depicted in Figure 6.3a.

We create the observation gadget for all existential variables x and for all (universal) variables $y \in O_x$, depicted in Figure 6.3b. For convenience, we write O for the pair (x, y) corresponding to observation of y by x.

Finally, we create the clause gadget, depicted in Figure 6.3c. A vertex γ_i certainly communicates with $\top_z$ iff $z \in \gamma_i$, and with $\bot_z$ iff $\neg z \in \gamma_i$. Moreover, the vertex v_γ communicates with all $\top_z$ and $\bot_z$ for all variables z.

We define the initial and target configurations as:

$$c^s = \langle s_\gamma, s_{\gamma_1}, \ldots, s_{\gamma_m}, s_{x_1}, \ldots, s_{x_n}, s_{y_1}, \ldots, s_{y_n}, s_{O_1}, \ldots, s_{O_k} \rangle, \text{ and}$$

$$c^g = \langle t_\gamma, t_{\gamma_1}, \ldots, t_{\gamma_m}, t_{x_1}, \ldots, t_{x_n}, t_{y_1}, \ldots, t_{y_n}, t_{O_1}, \ldots, t_{O_k} \rangle.$$

Lemma 6.5. *DQBF φ holds if and only if $(G_1, G_2, c^s, c^g, \infty)$ is positive.*

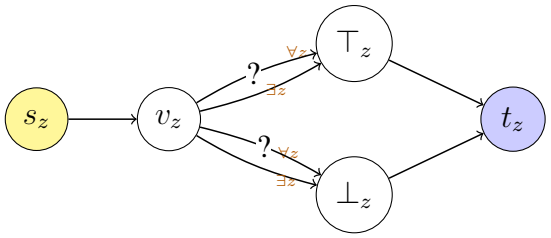
Proof. We denote the agents as such: (1) an agent that starts at s_z as the *existential agent* z if z is an existential variable; the agent is called *universal* if z is universal; (2) the *verification agent* is the one starting at s_γ; (3) the *clause agents* γ_i start at s_{γ_i}; (4) the *observation agents* O start at s_O.

($\Rightarrow$) Suppose the DQBF φ holds, and let A be the collection of Skolem functions. We build the following joint strategy. The environment chooses the truth values of universal variables z by deleting some edges v_z to $\top_z$ or v_z to $\bot_z$. If the environment deletes the edge v_z to $\top_z$, the agent is forced to pass in $\bot_z$, thus the variable z is considered to be false. If the environment deletes the edge v_z to $\bot_z$, the agent is forced to pass in $\top_z$, thus variable z is considered to be true. If the environment deletes neither edge, then we define the strategy for agent a_y to choose to pass in y, making y true by default.

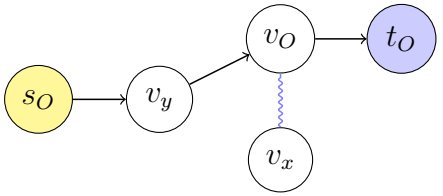
The rest of the strategy is defined as follows. At the first step, each variable agent for variable z moves to v_z, and each observation agent for the pair (x, y) moves to v_y, and thus observes the value of universal variable y. At the second step, existential agents remain in place, while observation agents move to v_O, thus sharing their observations with the corresponding existential agents. Thus, at this point, each existential agent corresponding to variable x knows the values of all universal variables $y \in O_x$. Then, agent z moves to $\top_z$ if $A_z(\nu) = 1$ and to $\bot_z$ otherwise, where ν is the valuation of the variables in O_z.

All clause agents move from s_i to γ_i and remain at γ_i for two steps so that all existential and universal variable agents z are at $\top_z$ or $\bot_z$. The verification moves to v_γ and also waits for two steps. Since each clause is satisfied by the currently read valuation, each clause agent γ_i communicates at least with one existential or universal agent. Thus, the verification agent communicates with *all* clause agents via these variable agents. Since the clause agents communicate with the verification agent at this moment, the latter can see which edges are present in the clause gadget, and can continue go to t_γ without getting stuck.

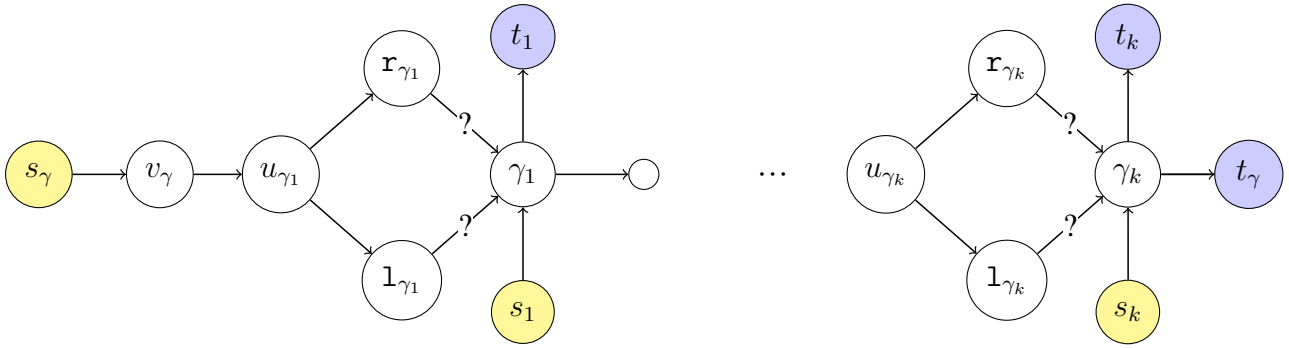
($\Leftarrow$) Conversely, suppose there is a witness joint strategy, in particular ensuring that the verification agent goes to t_z. This means that the agent must have received all the information about the topology around vertices $\gamma_1, \ldots, \gamma_k$. But this is only possible if the agents have occupied a configuration in which the verification agent is at v_γ, all clause agents are at γ_i such that for each clause γ_i, there is at least one variable agent z at $\top_z$ if $z \in \gamma_i$ and at $\bot_z$ if $\neg z \in \gamma_i$. Thus, φ is a positive instance of TDQBF. $\triangle$



(a) Gadget for variable z. Both edges $(v_z, \top_z)$ and $(v_z, \bot_z)$ are certain (resp. uncertain) if z is existential (resp. universal).



(b) Gadget for the observation O of universal variable y from existential variable x.



(c) Clause gadget.

⟶ Certain movement edge

〰 Certain communication edge

–?→ Uncertain movement edge

Certain (resp. uncertain) movement edge if z is existential (resp. universal)

$(\top_z, \gamma_i) \in E_1^c$ when z appears in γ_i
$(\bot_z, \gamma_i) \in E_1^c$ when $\neg z$ appears in γ_i
$(\top_z, v_\gamma) \in E_1^c, (\bot_z, v_\gamma) \in E_1^c$

Figure 6.3 – Gadgets in the reduction from DQBF to unbounded decentralized reachability.

□

6.5.2 Bounded Case

In the bounded case, the problem is NEXPTIME-complete independently of the encoding of the bound. Moreover, the hardness holds even for undirected graphs.

Theorem 6.5. ***bDMAPFI*** *is NEXPTIME-complete, NEXPTIME-hardness holds for undirected graphs.*

Proof. The upper bound when the bound k is given in unary is obtained by the following non-deterministic algorithm:

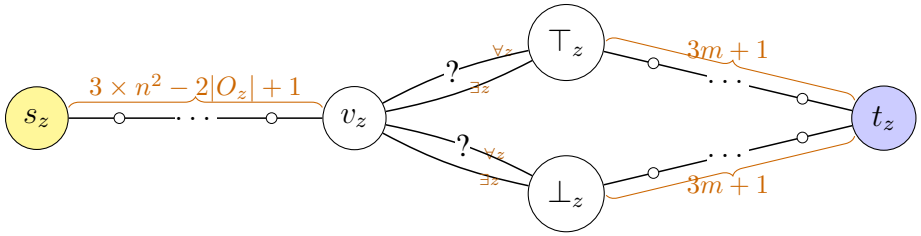
(1) Guess a strategy σ_i for each agent i, up to executions of length $\leq k$. Such a strategy can be represented as a tree of depth k, and thus has size exponential in k.

(2) Check that σ_i is uniform for agent i.

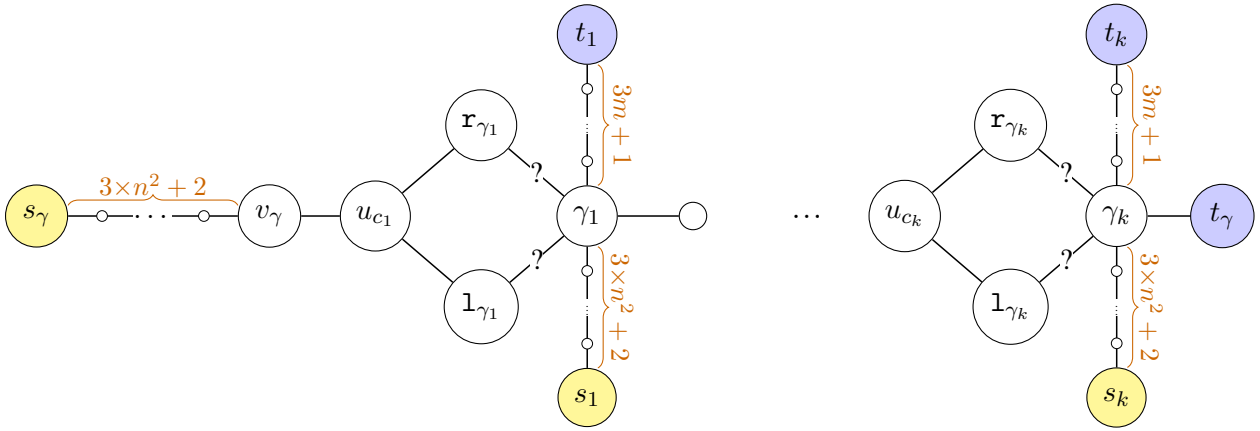
(3) For all admissible graphs G such that $G_1 \subseteq G \subseteq G_2$, execute the joint strategy σ and check that the outcome execution from the initial configuration leads to the target configuration.

The obtained algorithm is non-deterministic and runs in exponential time. Note that the encoding of k is not relevant since for $k \geq 2|V|$ there is always a solution following the unbounded case.

We now prove the NEXPTIME-hardness result by reduction from TDQBF. Given an instance of TDQBF $\forall y_1, \ldots, y_n \exists x_1(O_{x_1}) \ldots \exists x_n(O_{x_n})\ \psi$, we build an instance of bounded decentralized reachability (G_1, G_2, c^s, c^g, k). We denote the number of clauses by m.



(a) Variable gadget for variable z. Edges between v_z and $\top_z$, and between v_z and $\bot_z$ are both certain if z is existential and both uncertain if z is universal.



(b) Clauses gadget.

—— Certain Undirected Movement edge

–?– Uncertain Undirected Movement edge

Undirected Movement edge that is certain if z is existential and uncertain if z is universal

$(\top_z, \gamma_i) \in E_1^c$ when z appears in γ_i
$(\bot_z, \gamma_i) \in E_1^c$ when $\neg z$ appears in γ_i
$(\top_z, v_\gamma) \in E_1^c, (\bot_z, v_\gamma) \in E_1^c$

Figure 6.4 – Gadgets for the reduction from DQBF to the bounded decentralized reachability problem.

We construct the graph G_1 as follows:

For each variable z, we create a gadget, as depicted in Figure 6.4a. We create the vertices s_z, v_z, $\top_z$, $\bot_z$ and t_z, and we create a movement path of length $3m+1$ from t_z to $\top_z$ and $\bot_z$. Then, if the variable z is universal, we create a movement path of length $3 \times n^2 + 1$ from s_z to v_z. If the variable z is x_i, that is the i-th existential variable, we build a movement path of length $3 \times n^2 - 2|O_z| + 1$ from s_z to v_z as follows. We first build a movement path of length $3 \times (in - |O_z|)$. We then extend this movement by a vertex ρ_y for each $y \in O_z$. We extend our path from the last such vertex ρ_y to v_z by a movement path of length $3 \times (n^2 - in)$. Furthermore, we add a bidirectional edge between ρ_y and v_y.

We create the clause gadget, depicted in Figure 6.4b, composed of the vertices s_γ, v_γ, t_γ and for all clauses γ_i, the vertices u_{γ_i}, $\mathtt{l}_{\gamma_i}$, $\mathtt{r}_{\gamma_i}$, s_i, c_i and t_i. We create a movement path of length $3 \times n^2 + 2$ between s_γ and v_γ and between all s_i and γ_i. For all γ_i, we create movement edges from u_{γ_i} to $\mathtt{l}_{\gamma_i}$ and to $\mathtt{r}_{\gamma_i}$, from the vertex γ_i to $u_{\gamma_{i+1}}$, or t_γ if $i = m$. In addition, we add a movement edge between v_γ and u_{γ_1}. For all γ_i, we add a movement path of length $3m+1$ between vertices γ_i and t_i. Vertex γ_i communicates with $\top_z$ iff $z \in \gamma_i$, and with $\bot_z$ iff $\neg z \in \gamma_i$. Moreover, vertex v_γ communicates with all $\top_z$ and $\bot_z$ for all variables z.

Graph G_2 contains G_1 and for all universal variables z, we add movement edges: from v_z to $\top_z$ and to $\bot_z$, and for all clauses γ_i, we add movement edges from the vertex γ_i to $\mathtt{l}_{\gamma_i}$ and to $\mathtt{r}_{\gamma_i}$.

We define the initial and target configurations as $c^s = \langle s_\gamma, s_{\gamma_1}, \ldots, s_{\gamma_m}, s_{z_1}, \ldots, s_{z_n} \rangle$ and $c^g = \langle t_\gamma, t_{\gamma_1}, \ldots, t_{\gamma_m}, t_{z_1}, \ldots, t_{z_n} \rangle$.

We show that φ is a positive instance of TDQBF iff (G_1, G_2, c^s, c^g, k) is positive with $k = 3 \times n^2 + 2 + 3m + 1$. We refer to an agent that starts at s_z as the *existential agent* z if z is an existential variable; the agent is called *universal* if z is universal. The *verification agent* is the agent that starts at s_γ and the *clause agents* γ_i start at s_{γ_i}.

($\Rightarrow$) Suppose the DQBF holds, and let A be an assignment function. We build a joint strategy. The environment chooses the truth values of variables z by deleting some edges v_z to $\top_z$ or v_z to $\bot_z$. If the environment deletes the edge v_z to $\top_z$, it enforces the agent to pass in $\bot_z$, thus the variable z is considered to be false. If the environment deletes the edge v_z to $\bot_z$, it enforces the agent to pass in $\top_z$, thus variable z is considered to be true. If the environment deletes neither edge, then we define the strategy for agent a_y to choose to pass in y, making y true by default.

The strategy is defined as follows. Each existential agent z follows the movement path of length $3 \times n^2 - 2|O_z| + 1$ to v_z, but whenever they have vertex v_y as a neighbor, they visit v_y, come back, and continue their paths. This happens exactly O_z times, so at time $3 \times n^2 + 1$, the existential agent is at u_z. Note that along this path the agent has visited all vertices $v_{z'}$ with $z' \in O_z$, thus knows which edge among $(v_{z'}, \top_{z'})$ and $(v_{z'}, \bot_{z'})$ is present. Thus, upon arriving to v_z, the agent has the knowledge of the valuation for all variables in O_z. Observe also that by construction of the movement paths between s_z and v_z, the agents never meet in this phase of the execution. Agent z moves to $\top_z$ if $A_z(\nu) = 1$ and to $\bot_z$ otherwise, where ν is the valuation of the variables in O_z.

All clause agents reach γ_i at time $3 \times n^2 + 2$. Moreover, the verification agent is at v_γ at this point, and all existential and universal variable agents z are at $\top_z$ or $\bot_z$. Recall that all these

vertices communicate. Since each clause is satisfied by the currently read valuation, each clause agent γ_i communicates at least with one existential or universal agent. Thus, the verification agent communicates with *all* clause agents via these variable agents. Since the clause agents can see which edges are present in the clause gadget, the verification agent has now full information about this gadget, and can continue their path until t_γ without backtracking, thus in total time $3 \times n^2 + 2 + 3m + 1$.

($\Leftarrow$) Conversely, suppose there is a joint strategy enforcing that the verification agent goes to t_γ in k steps. Thus, it means that she must have received all the information about the surroundings of the vertices $\gamma_1, \ldots, \gamma_k$, as she has no time to backtrack from a wrong choice. This information can only be sent to the verification agent from the clause agents through the variable agents after $3 \times n^2 + 2$ steps. The variable agents representing the assignments are connecting (i.e. satisfying) all clauses. Indeed, for all γ_i, there is one variable agent z at $\top_z$ if $z \in \gamma_i$ and at $\bot_z$ if $\neg z \in \gamma_i$. Thus, the DQBF is a positive instance of TDQBF. □

6.6 Discussion

6.6.1 Additional Results

We present results obtained by a simple observation/modification.

Unbounded Reachability and Undirected Graphs. Both the unbounded connected and unbounded decentralized reachability become *trivial* on undirected graphs. This is because we only require reachability for (c-)admissible graphs. In the decentralized case, each agent can run a DFS independently, and eventually reach their targets in at most $2|V|$ steps, and a similar search can be done by the set of agents in the connected case.

Base Station. Several works consider a designated *base* vertex to which all agents must stay connected during the execution [151, 30, 28]. This concept is only relevant in the connected case. Our results also hold with this additional constraint. In fact, the lower bound of Theorem 6.1 follows from Tateo et al. [151], which proves the bound also with a base. In Theorem 6.3, we can add the base vertex as an isolated vertex so that the reduction is still valid.

Collisions. We did not require the paths to be collision-free in the results presented in this paper. However, this property is already ensured by our proofs or can be obtained by simple modifications. The lower bound proof of Theorem 6.1 relies on Theorem 3.2 from Tateo et al. [151] which holds with collision constraints as well, so this is also true for our case. The proof of Theorem 6.3 does not generate collision-free paths as the groups of occurrence agents start and finish at the same location and follow almost the same path. This proof can be adapted to prevent collisions by delaying each occurrence agent by 3 steps behind one another. This can be achieved easily by extending the movement paths and shifting the starting location and target location of an agent up by 3 vertices behind the previous agent. The proof of lower bound of Theorem 6.4 features a construction ensuring a collision-free strategy. Indeed, the observations agents only need to take turns to visit the universal variables. Thus, the result holds with collision constraints as well. The algorithms of Theorem 6.1, 6.3, and 6.5 can be adapted by restricting all considered configurations to collision-free ones; while c-admissibility

of a graph with collision constraints can be checked using Theorem 3.2.

Graph Classes. The MAPF and CMAPF problems have been studied for different classes of graphs (planar, grid, ...). The proof of lower bound in Theorem 6.1 relies on the proof of unbounded reachability done in Charrier et al. [28], thus the result of PSPACE-hardness on planar graphs also carries over to our problem. Planar QBF is known to be PSPACE-complete [95], and the construction of Theorem 6.3 is such that when applied to a planar QBF, the resulting graph is planar. Our PSPACE-completeness result thus holds on planar graphs.

6.6.2 Related Work

Different definitions of robust plans have been studied [103, 8, 9]. A k-*robust plan* guarantees the reachability of the target in the events of at most k delays. A p-*robust plan* executes without a conflict with probability at least p. Our framework does not consider delayed agents but focus on synchronous executions with incomplete knowledge of the area.

The problem of MAPF with a dynamic environment has multiple formulations. The Adversarial Cooperative Path-Finding [76] considers that the obstacles are agents which reason to prevent the cooperation to reach its goal. Murano et al. [111] considered the problem where the dynamics of the environment is predictable. Additionally, when obstacles have unknown dynamics, one can estimate their movements and plan to minimize the probability of a collision [109], or predict their movements [61] and plan online the movement of the agents [173]. In our setting, the environment is static, thus, all observations are fixed.

MAPF with Uncertainty (MAPFU) asks for a plan which guarantees that mishaps, localization and sensing errors do not impact the proper execution of the plan. This problem can be solved by temporal logic [156], POMDPs [108], replanning [146, 48, 96], interaction regions [39, 50], and belief space planning [21, 59, 125]. Nebel et al. [112] studied the MAPF problem with an uncertainty on the destination of the agents and lack of communication. The asynchronous movement of the agents, studied in those papers, cannot be expressed in our framework as we require the agent to follow some universal clock to execute their plan.

6.6.3 Perspectives

We proposed a setting for CMAPF in the incomplete case and studied the theoretical complexity of the reachability problem. The first natural question is to find classes of graphs (e.g. grid graphs) on which the reachability problem is easier to solve, as it was done for MAPF in Wang and Botea [160] and Banfi et al. [14], and CMAPF in [28]. Another possible direction is to study the coverage of all vertices [28]. An alternative way to handle non-admissible graphs is to require that agents return to their starting configuration if the graph is discovered not to be admissible. We believe that such variants should be as hard as reachability. Furthermore, there are several possible generalizations that could be considered by introducing dynamic environments (instead of static), faulty sensing of agents, robustness, uncertainty, etc.

CHAPTER 7

CONCLUSION

In this chapter, we provide a summary of the contributions of this dissertation. Then, we present the different articles that have been published during the writing of this work. Finally, we provide the reader with interesting future works and research directions which can follow this work.

7.1 Contributions

More autonomous agents are being used every day and the tasks they perform become increasingly more complex [56, 119, 39, 162]. Those applications require multiple agents to cooperate together towards a goal. As the tasks complexify, it is not practical to reason about the group of agents as a single entity. Thus, there is a need for algorithms that let agents communicate and be aware of each other, plan together, avoid mistakes, and "survive" mishaps.

While many works attempted to tackle the problem of planning with communication and/or in a partially-known environment, none of the existing work presented a framework to properly study the complexity and algorithmic methods of multi-agent path planning with a formal communication system. This dissertation has for purpose to fill this gap by providing an in-depth study of the connected multi-agent path planning problems with incomplete knowledge.

7.1.1 Connectivity

Part I introduced a framework which formalizes environment-based communication system for multi-agent path planning problems. This setting allowed us to study connected planning independently of the communication system used by the agents in Chapter 4. Note that, the applications of this framework define the communication, thus, the result of this work hold for most common communication systems (e.g. line-of-sight, radius). This first contribution provided a broad overview of the difficulties of planning with communication as well as some ways to ease the problems. In particular, we identify a class of graphs which correspond to realistic applications and reduces the complexity to LOGSPACE.

Chapter 5 presents an optimal algorithm to solve CMAPF which is able to, in some cases, outperform known approximate algorithms. We provided a proof of its optimality and soundness, and we showed that our algorithm may not terminate on unsatisfiable instances. We provided experimentation which shows that, while outperformed, our algorithm can offer a good alternative by scaling better in connected worlds and generating execution, by an order of magnitude shorter. Finally, as it is based on a similar structure as the well-studied CBS algorithm for

MAPF, one can extend it to support the generation of collision-free executions and may adapt some known optimizations.

7.1.2 Incomplete Knowledge

Secondly, we provided an extension of the framework presented in Chapter 4 which can model partially-known environments. The case of incomplete knowledge in multi-agent systems is crucial as almost no real world application can ensure a perfectly known world. Importantly, our framework offers a way to model the loss of communication and the reconnection of agents in a positional manner. We studied the impact of enforcing a permanent connectivity and allowing disconnections. This study gives an initial step towards understanding precisely the cost of communication in unknown worlds.

7.2 Publications

The majority of the content of this dissertation has been published in International peer-reviewed conferences and journals.

Conference:

1. T. Charrier, A. Queffelec, O. Sankur, F. Schwarzentruber (2019), *Reachability and Coverage Planning for Connected Agents: Extended Abstract*, International Conference on Autonomous Agents and Multi-Agent Systems.
2. T. Charrier, A. Queffelec, O. Sankur, F. Schwarzentruber (2019), *Reachability and Coverage Planning for Connected Agents*, International Joint Conference on Artificial Intelligence.
3. A. Queffelec, O. Sankur, F. Schwarzentruber (2021), *Planning for Connected Agents in a Partially Known Environment*, Canadian Conference on Artificial Intelligence.

Journal:

4. T. Charrier, A. Queffelec, O. Sankur, F. Schwarzentruber (2020), *Complexity of planning for connected agents*, Journal of Autonomous Agents and Multi-Agent Systems.

Demo:

5. A. Queffelec, O. Sankur, F. Schwarzentruber (2018), *Generating Plans for Cooperative Connected UAVs*, International Joint Conference on Artificial Intelligence.
6. T. Charrier, A. Queffelec, O. Sankur, F. Schwarzentruber (2021), *Connected Multi-Agent Path Finding: Generation and Visualization*, International Joint Conference on Artificial Intelligence.

The integrality of Chapter 4 has been published throughout publications 1,2,4. The study in Chapter 5 has been partially published in 5 and is under preparation. Finally, the publication 3 is a short version of Chapter 6.

7.3 Future Work

In order to conclude this dissertation we would like to give the reader a look beyond the content of our study. In particular, we wish to provide the reader with a view of the potential solutions to solve the high-complexity problems defined in this work.

Our first work, Chapter 4, gave a good overview of the complexity of CMAPF. However, our framework describes an arbitrary communication system, and one may be interested to study the complexity of a specific system (e.g. radius, line-of-sight). Indeed, as shown by the study of the sigh-moveable graphs, in Section 4.5, it may be that some communication systems are easier to plan with. For instance, our constructions in proofs often rely on fully connected locations (see proofs of Section 4.4), which may not be realistic. Furthremore, most of our proofs, even in planar cases, also rely on non-euclidean movement graphs (see proofs of Section 4.7). It may be the case that under a euclidean hypothesis, some problems are simpler, or admits powerful approximation algorithms.

Secondly, we provided a simple algorithmic solution to optimally solve CMAPF in Chapter 5. However, by taking inspiration from the optimizations of CBS, the scalability may be greatly improved. Furthermore, one may wish to combine CCBS with CBS to obtain the first algorithm capable of generating connected and collision-free execution, although it would require rethinking most optimizations of CCBS and CBS. Additionally, CMAPF seems to be a very cooperation-demanding problem, as disconnection is global to the group, a decoupled algorithm may not be the most efficient approach.

We showed that solving the connected variant of CMAPF in an incomplete environment is, in the worst case, as hard as solving CMAPF. However, the algorithmic solutions employed to solve CMAPF are not adaptable to the incomplete knowledge case. Indeed, an unknown environment requires the agents to be equipped with a strategy. Fortunately, related lines of works are that of the CTP and Partially Observable Markov Decision Processes (POMDPs), for which many algorithms have been developed and may be adapted [97, 49, 158, 63, 17].

Finally, while the PSPACE complexity of the previous problems may still be practical, it leaves little hope for solving the decentralized variant. Fortunately, our problem can be modeled as a Dec-POMDPs and may benefit from the state-of-the-art planners [38, 62, 6, 117]. Another approach to solve the decentralized case of CMAPF would be to use a Multi-Agent Reinforcement Learning (MARL) approach. Many recent works have been investigating the efficiency of RL in multi-agent systems [36, 37], see [124] for a comprehensive survey on robotic applications.

BIBLIOGRAPHY

[1] Ali Ahmadzadeh, James Keller, George Pappas, Ali Jadbabaie, and Vijay Kumar, « An Optimization-Based Approach to Time-Critical Cooperative Surveillance and Coverage with UAVs », *in*: *Experimental Robotics: The 10th International Symposium on Experimental Robotics*, Berlin, Heidelberg, 2008, pp. 491–500, DOI: 10.1007/978-3-540-77457-0_46.

[2] C. Amato, G. Chowdhary, A. Geramifard, N. K. Üre, and M. J. Kochenderfer, « Decentralized Control of Partially Observable Markov Decision Processes », *in*: *CDC*, 2013, pp. 2398–2405, DOI: 10.1109/CDC.2013.6760239.

[3] F. Amigoni, J. Banfi, and N. Basilico, « Multirobot Exploration of Communication-Restricted Environments: A Survey », *in*: *IEEE Intelligent Systems* 32.*6* (Nov. 2017), pp. 48–57, ISSN: 1941-1294, DOI: 10.1109/MIS.2017.4531226.

[4] F. Amigoni, V. Caglioti, and Umberto Galtarossa, « A Mobile Robot Mapping System with an Information-Based Exploration Strategy », *in*: *ICINCO*, 2004.

[5] S. P. Anbuudayasankar, K. Ganesh, and S. Mohapatra, *Models for practical routing problems in logistics*, 2016.

[6] Raghav Aras and Alain Dutech, « An Investigation into Mathematical Programming for Finite Horizon Decentralized POMDPs », *in*: 37.*1* (Jan. 2010), pp. 329–396, ISSN: 1076-9757.

[7] Sanjeev Arora and Boaz Barak, *Computational complexity: a modern approach*, 2009.

[8] Dor Atzmon, Ariel Felner, Roni Stern, Glenn Wagner, Roman Barták, and Neng-Fa Zhou, « k-Robust Multi-Agent Path Finding », *in*: *International Symposium on Combinatorial Search (SoCS)*, 2017, pp. 157–158.

[9] Dor Atzmon, Roni Stern, Ariel Felner, Nathan R. Sturtevant, and Sven Koenig, « Probabilistic Robust Multi-Agent Path Finding », *in*: *Proc. of ICAPS*, 2020, pp. 29–37.

[10] Dor Atzmon, Roni Stern, Ariel Felner, Nathan R. Sturtevant, and Sven Koenig, « Probabilistic Robust Multi-Agent Path Finding », *in*: *Proceedings of the International Conference on Automated Planning and Scheduling* 30.*1* (June 2020), pp. 29–37.

[11] Dor Atzmon, Roni Stern, Ariel Felner, Glenn Wagner, Roman Barták, and Neng-Fa Zhou, « Robust multi-agent path finding », *in*: *Eleventh Annual Symposium on Combinatorial Search*, 2018.

[12] Franz Aurenhammer, « Voronoi Diagrams - A Survey of a Fundamental Geometric Data Structure », *in*: *ACM Comput. Surv.* 23.*3* (1991), pp. 345–405, DOI: 10.1145/116873.116880.

[13] J. Banfi, A. Q. Li, N. Basilico, I. Rekleitis, and F. Amigoni, « Asynchronous multi-robot exploration under recurrent connectivity constraints », *in*: *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5491–5498, DOI: 10.1109/ICRA.2016.7487763.

[14] Jacopo Banfi, Nicola Basilico, and Francesco Amigoni, « Intractability of Time-Optimal Multirobot Path Planning on 2D Grid Graphs with Holes », *in*: *IEEE Robotics and Automation Letters* 2.*4* (2017), pp. 1941–1947, DOI: 10.1109/LRA.2017.2715406.

[15] Amotz Bar-Noy and Baruch Schieber, « The Canadian Traveller Problem. », *in*: *SODA*, vol. 91, 1991, pp. 261–270.

[16] Raphaël Berthon, Bastien Maubert, Aniello Murano, Sasha Rubin, and Moshe Y. Vardi, « Strategy logic with imperfect information », *in*: *LICS*, 2017, pp. 1–12, DOI: 10.1109/LICS.2017.8005136, eprint: 1805.12592.

[17] Zahy Bnaya, Ariel Felner, Dror Fried, Olga Maksin, and Solomon Shimony, « Repeated-Task Canadian Traveler Problem », *in*: vol. 28, Jan. 2011, DOI: 10.3233/AIC-150665.

[18] François Bodin, Tristan Charrier, Arthur Queffelec, and François Schwarzentruber, « Generating Plans for Cooperative Connected UAVs », *in*: *IJCAI 2018*, July 2018, pp. 5811–5813, DOI: 10.24963/ijcai.2018/846.

[19] Eli Boyarski, Ariel Felner, Guni Sharon, and Roni Stern, « Don't Split, Try To Work It Out: Bypassing Conflicts in Multi-Agent Pathfinding », *in*: *ICAPS 2015*, 2015, pp. 47–51.

[20] Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, David Tolpin, Oded Betzalel, and Solomon Eyal Shimony, « ICBS: Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding », *in*: *IJCAI 2015*, 2015, pp. 740–746.

[21] A. Bry and N. Roy, « Rapidly-exploring Random Belief Trees for motion planning under uncertainty », *in*: *Proc. of ICRA* (2011), pp. 723–730.

[22] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, « Collaborative multi-robot exploration », *in*: *Proc. of ICRA*, vol. 1, 2000, pp. 476–481, DOI: 10.1109/ROBOT.2000.844100.

[23] W. Burgard, M. Moors, C. Stachniss, and F. E. Schneider, « Coordinated multi-robot exploration », *in*: *IEEE Transactions on Robotics* 21.*3* (2005), pp. 376–386, DOI: 10.1109/TRO.2004.839232.

[24] Tom Bylander, « The Computational Complexity of Propositional STRIPS Planning », *in*: *Artif. Intell.* 69.*1-2* (1994), pp. 165–204, DOI: 10.1016/0004-3702(94)90081-7.

[25] Tauã M. Cabreira, Lisane B. Brisolara, and Paulo R. Ferreira Jr., « Survey on Coverage Path Planning with Unmanned Aerial Vehicles », *in*: *Drones* 3.*1* (2019), ISSN: 2504-446X, DOI: 10.3390/drones3010004.

[26] K. Cesare, R. Skeele, Soo-Hyun Yoo, Yawei Zhang, and G. Hollinger, « Multi-UAV exploration with limited communication and battery », *in*: *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 2230–2235, DOI: 10.1109/ICRA.2015.7139494.

[27] Ashok K. Chandra, Dexter Kozen, and Larry J. Stockmeyer, « Alternation », *in*: *J. of ACM* 28.*1* (1981), pp. 114–133, DOI: 10.1145/322234.322243.

[28] Tristan Charrier, Arthur Queffelec, Ocan Sankur, and François Schwarzentruber, « Complexity of planning for connected agents », *in*: *Auton. Agents Multi Agent Syst.* 34.*2* (2020), p. 44, DOI: 10.1007/s10458-020-09468-5.

[29] Tristan Charrier, Arthur Queffelec, Ocan Sankur, and François Schwarzentruber, « Reachability and Coverage Planning for Connected Agents », *in*: *Proceedings of AAMAS, Montreal, QC, Canada, May 13-17, 2019*, 2019, pp. 1874–1876.

[30] Tristan Charrier, Arthur Queffelec, Ocan Sankur, and François Schwarzentruber, « Reachability and Coverage Planning for Connected Agents », *in*: *IJCAI 2019*, 2019, pp. 144–150, DOI: 10.24963/ijcai.2019/21.

[31] Y. Chen, H. Zhang, and M. Xu, « The coverage problem in UAV network: A survey », *in*: *Fifth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, July 2014, DOI: 10.1109/ICCCNT.2014.6963085.

[32] Howie Choset, « Coverage for robotics - A survey of recent results », *in*: *Ann. Math. Artif. Intell.* 31.*1-4* (2001), pp. 113–126, DOI: 10.1023/A:1016639210559.

[33] Howie Choset and Philippe Pignon, « Coverage Path Planning: The Boustrophedon Cellular Decomposition », *in*: *Field and Service Robotics*, London, 1998, pp. 203–209, ISBN: 978-1-4471-1273-0.

[34] Stephen A. Cook, « A Taxonomy of Problems with Fast Parallel Algorithms », *in*: *Information and Control* 64.*1-3* (1985), pp. 2–21, DOI: 10.1016/S0019-9958(85)80041-3, URL: https://doi.org/10.1016/S0019-9958(85)80041-3.

[35] T. Danner and L. E. Kavraki, « Randomized planning for short inspection paths », *in*: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, Apr. 2000, 971–976 vol.2, DOI: 10.1109/ROBOT.2000.844726.

[36] Yann-Michaël De Hauwere, Sam Devlin, Daniel Kudenko, and Ann Nowé, « Context-sensitive reward shaping for sparse interaction multi-agent systems », *in*: *The Knowledge Engineering Review* 31.*1* (2016), pp. 59–76.

[37] Sam Devlin and Daniel Kudenko, « Theoretical Considerations of Potential-Based Reward Shaping for Multi-Agent Systems », *in*: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '11, 2011, pp. 225–232, ISBN: 0982657153.

[38] Jilles Steeve Dibangoye, Christopher Amato, Olivier Buffet, and François Charpillet, « Optimally solving Dec-POMDPs as continuous-state MDPs », *in*: *Journal of Artificial Intelligence Research* 55 (2016), pp. 443–497.

[39] Kurt Dresner and Peter Stone, « A Multiagent Approach to Autonomous Intersection Management », *in*: *JAIR* 31.*1* (2008), pp. 591–656, DOI: 10.1613/jair.2502.

[40] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, « Robotic exploration as graph construction », *in*: *IEEE Transactions on Robotics and Automation* 7.*6* (1991), pp. 859–865, DOI: 10.1109/70.105395.

[41] Brendan Englot and Franz Hover, « Planning Complex Inspection Tasks Using Redundant Roadmaps », *in*: *Robotics Research : The 15th International Symposium ISRR*, Cham, 2017, pp. 327–343, ISBN: 978-3-319-29363-9, DOI: 10.1007/978-3-319-29363-9_19.

[42] Brendan Englot and Franz Hover, « Sampling-Based Coverage Path Planning for Inspection of Complex Structures », *in*: *International Conference on Automated Planning and Scheduling*, 2012, URL: https://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4728.

[43] Esra Erdem, Doga G. Kisa, Umut Oztok, and Peter Schüller, « A General Formal Framework for Pathfinding Problems with Multiple Agents », *in*: *Proc. of AAAI*, 2013, pp. 290–296.

[44] Patrick Eyerich, Thomas Keller, and Malte Helmert, « High-Quality Policies for the Canadian Traveler's Problem. », *in*: vol. 1, Jan. 2010.

[45] Ariel Felner, Meir Goldenberg, Guni Sharon, Roni Stern, Tal Beja, Nathan Sturtevant, Jonathan Schaeffer, and Robert C. Holte, « Partial-expansion A* with Selective Node Generation », *in*: *AAAI 2012*, Toronto, Ontario, Canada, 2012, pp. 471–477.

[46] Ariel Felner, Jiaoyang Li, Eli Boyarski, Hang Ma, Liron Cohen, T. K. Satish Kumar, and Sven Koenig, « Adding Heuristics to Conflict-Based Search for Multi-Agent Path Finding », *in*: *ICAPS 2018*, vol. 28, 1, 2018, pp. 83–87.

[47] Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek, « Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges », *in*: *SoCS*, 2017, pp. 28–37.

[48] D. Ferguson, N. Kalra, and A. Stentz, « Replanning with RRTs », *in*: *Proc. of ICRA*, 2006, pp. 1243–1248, DOI: 10.1109/ROBOT.2006.1641879.

[49] D. Ferguson, A. Stentz, and S. Thrun, « PAO for planning with hidden state », *in*: *IEEE International Conference on Robotics and Automation, 2004*, vol. 3, 2004, 2840–2847 Vol.3, DOI: 10.1109/ROBOT.2004.1307491.

[50] Carlo Ferrari, Enrico Pagello, Jun Ota, and Tamio Arai, « Multirobot motion coordination in space and time », *in*: *Robotics and Autonomous Systems* 25.*3-4* (1998), pp. 219–229, DOI: 10.1016/S0921-8890(98)00051-7.

[51] Raphael A. Finkel and Jon Louis Bentley, « Quad Trees: A Data Structure for Retrieval on Composite Keys », *in*: *Acta Inf.* 4 (1974), pp. 1–9, DOI: 10.1007/BF00288933.

[52] G. Francès, M. Ramírez, N. Lipovetzky, and H. Geffner, « Purely Declarative Action Descriptions are Overrated: Classical Planning with Simulators », *in*: *Twenty-Sixth International Joint Conference on Artificial Intelligence,* 2017, DOI: 10.24963/ijcai.2017/600.

[53] Mengyu Fu, Alan Kuntz, Oren Salzman, and Ron Alterovitz, « Toward Asymptotically-Optimal Inspection Planning Via Efficient Near-Optimal Graph Search », *in*: *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019, DOI: 10.15607/RSS.2019.XV.057, eprint: 1907.00506.

[54] Enric Galceran and Marc Carreras, « A survey on coverage path planning for robotics », *in*: *Robotics and Autonomous Systems* 61.*12* (2013), pp. 1258–1276, ISSN: 0921-8890, DOI: https://doi.org/10.1016/j.robot.2013.09.004.

[55] Michael R Garey and David S Johnson, « Computers and intractability », *in*: *A Guide to the* (1979).

[56] Keivan Ghoseiri, Ferenc Szidarovszky, and Mohammad Jawad Asgharpour, « A multi-objective train scheduling model and solution », *in*: *Transportation Research Part B: Methodological* 38.*10* (2004), pp. 927–952, ISSN: 0191-2615, DOI: https://doi.org/10.1016/j.trb.2004.02.004.

[57] Dani Goldberg and Maja J. Matarić, « Interference as a Tool for Designing and Evaluating Multi-Robot Controllers », *in*: *Proc. of AAAI*, 1997, pp. 637–642.

[58] Meir Goldenberg, Ariel Felner, Roni Stern, Guni Sharon, and Jonathan Schaeffer, « A* variants for optimal multi-agent pathfinding », *in*: *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[59] J. P. Gonzalez and A. Stentz, « Planning with uncertainty in position an optimal and efficient planner », *in*: *IROS*, 2005, pp. 2435–2442, DOI: 10.1109/IROS.2005.1545048.

[60] Héctor González-Baños and Jean-Claude Latombe, « Navigation Strategies for Exploring Indoor Environments », *in*: *I. J. Robotic Res.* 21 (Oct. 2002), pp. 829–848, DOI: 10.1177/027836402128964099.

[61] N. C. Griswold and J. Eem, « Control for mobile robots in the presence of moving objects », *in*: *IEEE Trans. on Rob. and Autom.* 6.*2* (1990), pp. 263–268, DOI: 10.1109/70.54744.

[62] Eric A. Hansen, Daniel S. Bernstein, and Shlomo Zilberstein, « Dynamic Programming for Partially Observable Stochastic Games », *in*: AAAI'04, 2004, pp. 709–715, ISBN: 0262511835.

[63] Eric A. Hansen and Shlomo Zilberstein, « LAO*: A heuristic search algorithm that finds solutions with loops », *in*: *Artificial Intelligence* 129.*1* (2001), pp. 35–62, ISSN: 0004-3702, DOI: https://doi.org/10.1016/S0004-3702(01)00106-0.

[64] Peter E Hart, Nils J Nilsson, and Bertram Raphael, « A formal basis for the heuristic determination of minimum cost paths », *in*: *IEEE Transactions on Systems Science and Cybernetics,* (1968), pp. 100–107.

[65] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael, « Correction to "A Formal Basis for the Heuristic Determination of Minimum Cost Paths" », *in*: *SIGART Bull. 37* (Dec. 1972), pp. 28–29, ISSN: 0163-5719, DOI: 10.1145/1056777.1056779.

[66] N. Hazon and G. A. Kaminka, « Redundancy, Efficiency and Robustness in Multi-Robot Coverage », *in*: *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Apr. 2005, pp. 735–741, DOI: 10.1109/ROBOT.2005.1570205.

[67] Robert A. Hearn and Erik D. Demaine, « PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation », *in*: *Theoretical Computer Science* 343.*1* (2005), Game Theory Meets Theoretical Computer Science, pp. 72–96, ISSN: 0304-3975, DOI: https://doi.org/10.1016/j.tcs.2005.05.008.

[68] G. A. Hollinger and S. Singh, « Multirobot Coordination With Periodic Connectivity: Theory and Experiments », *in*: *IEEE Transactions on Robotics,* (Aug. 2012), pp. 967–973.

[69] Geoffrey A. Hollinger and Sanjiv Singh, « Multirobot Coordination With Periodic Connectivity: Theory and Experiments », *in*: *IEEE Trans. Robotics* 28.*4* (2012), pp. 967–973, DOI: 10.1109/TRO.2012.2190178.

[70] Steven Homer and Alan L Selman, *Computability and complexity theory*, 2011.

[71] J.E. Hopcroft, J.T. Schwartz, and M. Sharir, « On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE- Hardness of the "Warehouseman's Problem" », *in*: *The International Journal of Robotics Research* 3.*4* (1984), pp. 76–88, DOI: 10.1177/027836498400300405, eprint: https://doi.org/10.1177/027836498400300405.

[72] Andrew Howard and Nicholas Roy, *The Robotics Data Set Repository (Radish)*, 2003.

[73] Univ Rennes 1 / IRISA, *UAV RETINA Project*, http://eole-eyes.irisa.fr/, 2018.

[74] A. Itai and H. Shachnai, « Adaptive source routing in high-speed networks », *in*: *ISTCS*, June 1993, pp. 212–221, DOI: 10.1109/ISTCS.1993.253468.

[75] Alon Itai, Christos H. Papadimitriou, and Jayme Luiz Szwarcfiter, « Hamilton Paths in Grid Graphs », *in*: *SIAM J. Comput.* 11.*4* (1982), pp. 676–686, DOI: 10.1137/0211056.

[76] M. Ivanová and P. Surynek, « Adversarial Cooperative Path-Finding: Complexity and Algorithms », *in*: *ICTAI*, 2014, pp. 75–82, DOI: 10.1109/ICTAI.2014.22.

[77] M. Renee Jansen and Nathan R. Sturtevant, « Direction Maps for Cooperative Pathfinding », *in*: *Proceedings of the Fourth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'08, Stanford, California, 2008, pp. 185–190.

[78] R. M. Karp, « Reducibility Among Combinatorial Problems », *in*: *Proceedings of a symposium on the Complexity of Computer Computations,* 1972.

[79] L. E. Kavraki, M. N. Kolountzakis, and J. Latombe, « Analysis of probabilistic roadmaps for path planning », *in*: *Proceedings of IEEE International Conference on Robotics and Automation*, vol. 4, Apr. 1996, 3020–3025 vol.4, DOI: 10.1109/ROBOT.1996.509171.

[80] Philip N. Klein, Shay Mozes, and Oren Weimann, « Shortest Paths in Directed Planar Graphs with Negative Lengths: A Linear-space O(N Log2 N)-time Algorithm », *in*: *ACM Trans. Algorithms* 6.*2* (2010), 30:1–30:18, ISSN: 1549-6325, DOI: 10.1145/1721837.1721846.

[81] Aaron Knoll, « A survey of octree volume rendering methods », *in*: *GI, the Gesellschaft für Informatik*, 2006, p. 87.

[82] S. Koenig, C. Tovey, and W. Halliburton, « Greedy Mapping of Terrain », *in*: *Proc. of ICRA*, vol. 4, 2001, 3594–3599 vol.4, DOI: 10.1109/ROBOT.2001.933175.

[83] J. J. Kuffner and S. M. LaValle, « RRT-connect: An efficient approach to single-query path planning », *in*: *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, vol. 2, 2000, 995–1001 vol.2, DOI: 10.1109/ROBOT.2000.844730.

[84] Benjamin Kuipers and Yung-Tai Byun, « A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations », *in*: *Robotics and Autonomous Systems* 8.*1* (1991), Special Issue Toward Learning Robots, pp. 47–63, ISSN: 0921-8890, DOI: https://doi.org/10.1016/0921-8890(91)90014-C.

[85] Tushar Kusnur, Shohin Mukherjee, Dhruv Mauria Saxena, Tomoya Fukami, Takayuki Koyama, Oren Salzman, and Maxim Likhachev, « A Planning Framework for Persistent, Multi-UAV Coverage with Global Deconfliction », *in*: *CoRR* abs/1908.09236 (2019), arXiv: 1908.09236.

[86] Jean-Claude Latombe, *Robot motion planning*, vol. 124, 2012.

[87] Steven M. LaValle, *Planning Algorithms*, 2006, DOI: 10.1017/CBO9780511546877.

[88] Tod S. Levitt and Daryl T. Lawton, « Qualitative navigation for mobile robots », *in*: *Artificial Intelligence* 44.*3* (1990), pp. 305–360, ISSN: 0004-3702, DOI: https://doi.org/10.1016/0004-3702(90)90027-W.

[89] Harry R Lewis and Christos H Papadimitriou, « Elements of the Theory of Computation », *in*: *ACM SIGACT News* 29.*3* (1998), pp. 62–78.

[90] Jiaoyang Li, Ariel Felner, Eli Boyarski, Hang Ma, and Sven Koenig, « Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search », *in*: *IJCAI*, Aug. 2019, pp. 442–449, DOI: 10.24963/ijcai.2019/63.

[91] Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Ariel Felner, Hang Ma, and Sven Keonig, « Disjoint Splitting for Conflict-Based Search for Multi-Agent Path Finding », *in*: *ICAPS 2019*, 2019, pp. 279–283.

[92] Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Hang Ma, and Sven Koenig, « Symmetry-Breaking Constraints for Grid-Based Multi-Agent Path Finding », *in*: *SOCS*, 2019.

[93] Jiaoyang Li, Kexuan Sun, Hang Ma, Ariel Felner, TK Satish Kumar, and Sven Koenig, « Moving agents in formation in congested environments », *in*: *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 726–734.

[94] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W. Durham, T. K. Satish Kumar, and Sven Koenig, « Lifelong Multi-Agent Path Finding in Large-Scale Warehouses », *in*: *Proceedings of the 19th International Conference on Autonomous Agents and Multi-Agent Systems*, AAMAS '20, Auckland, New Zealand, 2020, pp. 1898–1900, ISBN: 9781450375184.

[95] D. Lichtenstein, « Planar Formulae and Their Uses », *in*: *SIAM Journal on Computing (SICOMP)* 11 (1982), pp. 329–343, DOI: 10.1137/0211025.

[96] Maxim Likhachev, Dave Ferguson, Geoff Gordon, Anthony Stentz, and Sebastian Thrun, « Anytime Dynamic A*: An Anytime, Replanning Algorithm », *in*: *Proc. of ICAPS*, 2005, pp. 262–271.

[97] Maxim Likhachev and Anthony Stentz, « Probabilistic planning with clear preferences on missing information », *in*: *Artificial Intelligence* 173.5 (2009), Advances in Automated Plan Generation, pp. 696–721, ISSN: 0004-3702, DOI: https://doi.org/10.1016/j.artint.2008.10.014.

[98] Lucian Vlad Lita, Jamieson Schulte, and Sebastian Thrun, « A System for Multi-Agent Coordination in Uncertain Environments », *in*: *Proc. of AGENTS*, 2001, pp. 21–22, DOI: 10.1145/375735.375806.

[99] Hang Ma, « Target Assignment and Path Planning for Navigation Tasks with Teams of Agents », PhD thesis, University of Southern California, 2020.

[100] Hang Ma, Wolfgang Hönig, TK Satish Kumar, Nora Ayanian, and Sven Koenig, « Lifelong path planning with kinematic constraints for multi-agent pickup and delivery », *in*: *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 01, 2019, pp. 7651–7658.

[101] Hang Ma and Sven Koenig, « AI Buzzwords Explained: Multi-Agent Path Finding (MAPF) », *in*: *AI Matters* 3 (2017), DOI: 10.1145/3137574.3137579.

[102] Hang Ma and Sven Koenig, « Optimal Target Assignment and Path Finding for Teams of Agents », *in*: *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems, Singapore, May 9-13, 2016*, 2016, pp. 1144–1152.

[103] Hang Ma, T. K. Satish Kumar, and Sven Koenig, « Multi-Agent Path Finding with Delay Probabilities », *in*: *Proc. of AAAI*, 2017, pp. 3605–3612.

[104] Hang Ma, Jiaoyang Li, TK Kumar, and Sven Koenig, « Lifelong multi-agent path finding for online pickup and delivery tasks », *in*: *arXiv preprint arXiv:1705.10868* (2017).

[105] Hang Ma, Craig A. Tovey, Guni Sharon, T. K. Satish Kumar, and Sven Koenig, « Multi-Agent Path Finding with Payload Transfers and the Package-Exchange Robot-Routing Problem », *in*: *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, 2016, pp. 3166–3173.

[106] Hang Ma, Jingxing Yang, L. Cohen, T. K. S. Kumar, and Sven Koenig, « Feasibility Study: Moving Non-Homogeneous Teams in Congested Video Game Environments », *in*: *AIIDE*, 2017.

[107] Laëtitia Matignon, Laurent Jeanpierre, and Abdel-Illah Mouaddib, « Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes », *in*: *Proc. of AAAI*, vol. 26, 1, 2012.

[108] S. A. Miller, Z. A. Harris, and E. K. P. Chong, « Coordinated Guidance of Autonomous UAVs via Nominal Belief-State Optimization », *in*: *ACC*, 2009, pp. 2811–2818, DOI: 10.1109/ACC.2009.5159963.

[109] Jun Miura and Y. Shirai, « Probabilistic Uncertainty Modeling of Obstacle Motion for Robot Motion Planning », *in*: *Journal of Robotics and Mechatronics* 14 (2002), pp. 349–356.

[110] Robert Morris, Corina S. Pasareanu, Kasper Søe Luckow, Waqar Malik, Hang Ma, T. K. Satish Kumar, and Sven Koenig, « Planning, Scheduling and Monitoring for Airport Surface Operations », *in*: *Planning for Hybrid Systems, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 13, 2016*, vol. WS-16-12, AAAI Workshops, 2016.

[111] A. Murano, Giuseppe Perelli, and S. Rubin, « Multi-agent Path Planning in Known Dynamic Environments », *in*: *PRIMA*, 2015, DOI: 10.1007/978-3-319-25524-8_14.

[112] Bernhard Nebel, Thomas Bolander, Thorsten Engesser, and Robert Mattmüller, « Implicitly Coordinated Multi-Agent Path Finding under Destination Uncertainty: Success Guarantees and Computational Complexity », *in*: *JAIR* 64.*1* (2019), pp. 497–527, ISSN: 1076-9757, DOI: 10.1613/jair.1.11376.

[113] Thomas Nestmeyer, Paolo Robuffo Giordano, Heinrich H. Bülthoff, and Antonio Franchi, « Decentralized Simultaneous Multi-Target Exploration Using a Connected Network of Multiple Robots », *in*: *Auton. Robots* 41.*4* (2017), pp. 989–1011, ISSN: 0929-5593, DOI: 10.1007/s10514-016-9578-9.

[114] Van Nguyen, Philipp Obermeier, Tran Cao Son, Torsten Schaub, and William Yeoh, « Generalized target assignment and path finding using answer set programming », *in*: *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[115] Evdokia Nikolova and David R. Karger, « Route Planning under Uncertainty: The Canadian Traveller Problem », *in*: *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2*, AAAI'08, Chicago, Illinois, 2008, pp. 969–974, ISBN: 9781577353683.

[116] Andreas Nüchter, Hartmut Surmann, and Joachim Hertzberg, « Planning Robot Motion for 3D Digitalization of Indoor Environments », *in*: *In Proc. of the 11th International Conference on Advanced Robotics (ICAR*, 2003, pp. 222–227.

[117] Frans A. Oliehoek, Matthijs T. J. Spaan, Christopher Amato, and Shimon Whiteson, « Incremental Clustering and Expansion for Faster Optimal Planning in Decentralized POMDPs », *in*: *J. Artif. Int. Res.* 46.*1* (Jan. 2013), pp. 449–509, ISSN: 1076-9757.

[118] Michael Otte and Nikolaus Correll, « Any-Com Multi-robot Path-Planning with Dynamic Teams: Multi-robot Coordination under Communication Constraints », *in*: *Experimental Robotics: The 12th International Symposium on Experimental Robotics*, Berlin, Heidelberg, 2014, pp. 743–757, ISBN: 978-3-642-28572-1, DOI: 10.1007/978-3-642-28572-1_51.

[119] Lucia Pallottino, Vincenzo G. Scordio, Antonio Bicchi, and Emilio Frazzoli, « Decentralized Cooperative Policy for Conflict Resolution in Multivehicle Systems », *in*: *IEEE Transactions on Robotics* 23.*6* (2007), pp. 1170–1183, DOI: 10.1109/TRO.2007.909810.

[120] R. Pandey, A. K. Singh, and K. M. Krishna, « Multi-robot exploration with communication requirement to a moving base station », *in*: *2012 IEEE International Conference on Automation Science and Engineering (CASE)*, Aug. 2012, pp. 823–828, DOI: 10.1109/CoASE.2012.6386475.

[121] Christos H Papadimitriou and Mihalis Yannakakis, « Shortest paths without a map », *in*: *Theoretical Computer Science* 84.*1* (1991), pp. 127–150, DOI: 10.1016/0304-3975(91)90263-2.

[122] Gary Peterson, John Reif, and Salman Azhar, « Lower Bounds for Multiplayer Noncooperative Games of Incomplete Information », *in*: *Comput & Math. Appl.* 41.*7-8* (2001), pp. 957–992, DOI: 10.1016/S0898-1221(00)00333-3.

[123] H. X. Pham, H. M. La, D. Feil-Seifer, and M. C. Deans, « A Distributed Control Framework of Multiple Unmanned Aerial Vehicles for Dynamic Wildfire Tracking », *in*: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2018), pp. 1–12, ISSN: 2168-2232, DOI: 10.1109/TSMC.2018.2815988.

[124] Athanasios S Polydoros and Lazaros Nalpantidis, « Survey of model-based reinforcement learning: Applications on robotics », *in*: *Journal of Intelligent & Robotic Systems* 86.*2* (2017), pp. 153–173.

[125] Samuel Prentice and Nicholas Roy, « The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance », *in*: *IJRR* 28 (Oct. 2009), pp. 1448–1465, DOI: 10.1177/0278364909341659.

[126] N SV Rao, Srikumar Kareti, Weimin Shi, and S Sitharama Iyengar, *Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms*, tech. rep., Oak Ridge National Lab., TN (United States), 1993.

[127] N. Rao, S. Kareti, Weimin Shi, and S. Iyengar, « Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms », *in*: 1993.

[128] Daniel Ratner and Manfred K. Warmuth, « Finding a Shortest Solution for the N × N Extension of the 15-PUZZLE Is Intractable », *in*: *Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, PA, USA, August 11-15, 1986. Volume 1: Science*, 1986, pp. 168–172.

[129] John H Reif, « Complexity of the mover's problem and generalizations », *in*: *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, IEEE Computer Society, 1979, pp. 421–427.

[130] Omer Reingold, « Undirected connectivity in log-space », *in*: *J. ACM* 55.*4* (2008), 17:1–17:24, DOI: 10.1145/1391289.1391291.

[131] Ioannis Rekleitis, Ai Peng New, Edward Samuel Rankin, and Howie Choset, « Efficient Boustrophedon Multi-Robot Coverage: an algorithmic approach », *in*: *Annals of Mathematics and Artificial Intelligence* 52.*2* (2008), pp. 109–142, ISSN: 1573-7470, DOI: 10.1007/s10472-009-9120-2.

[132] Martijn N. Rooker and Andreas Birk, « Multi-robot exploration under the constraints of wireless networking », *in*: *Control Engineering Practice* 15.*4* (2007), pp. 435–445, ISSN: 0967-0661, DOI: https://doi.org/10.1016/j.conengprac.2006.08.007.

[133] Malcolm R. K. Ryan, « Exploiting Subgraph Structure in MultiRobot Path Planning », *in*: *in Journal of Artificial Intelligence Research* 31 (2008), pp. 497–542, eprint: 1111.0053.

[134] Walter J. S., « Relationships Between Nondeterministic and Deterministic Tape Complexities », *in*: *Journal of Computer and System Sciences* (1970), DOI: 10.1016/S0022-0000(70)80006-X.

[135] Miguel Schneider-Fontán and M. Mataric, « Territorial multi-robot task division », *in*: *IEEE Trans. on Rob. and Autom.* 14 (1998), pp. 815–822.

[136] Jacob T. Schwartz and Micha Sharir, « On the Piano Movers' Problem: III. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving Amidst Polygonal Barriers », *in*: *The International Journal of Robotics Research* 2.*3* (1983), pp. 46–75, DOI: 10.1177/027836498300200304.

[137] Guni Sharon, Roni Stern, Ariel Felner, and Nathan Sturtevant, « Conflict-Based Search for Optimal Multi-agent Path Finding », *in*: *AAAI 2012*, Toronto, Ontario, Canada, 2012, pp. 563–569.

[138] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant, « Meta-Agent Conflict-Based Search For Optimal Multi-Agent Path Finding. », *in*: *SoCS* 1 (2012), pp. 39–40.

[139] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant, « Conflict-based search for optimal multi-agent pathfinding », *in*: *Artif. Intell.* 219 (2015), pp. 40–66, DOI: 10.1016/j.artint.2014.11.006.

[140] Davood Shiri and F. Sibel Salman, « On the Online Multi-Agent O—D k-Canadian Traveler Problem », *in*: *J. of Comb. Opt.* 34.*2* (2017), pp. 453–461, DOI: 10.1007/s10878-016-0079-8.

[141] David Silver, « Cooperative Pathfinding », *in*: *AIIDE 2005*, Marina del Rey, California, 2005, pp. 117–122.

[142] David Silver, « Cooperative Pathfinding », *in*: *Proceedings of the First AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, AIIDE'05, Marina del Rey, California, 2005, pp. 117–122.

[143] M. Sipser, *Introduction to the theory of computation*, 1997, ISBN: 978-0-534-94728-6.

[144] Kiril Solovey and Dan Halperin, « On the hardness of unlabeled multi-robot motion planning », *in*: *The International Journal of Robotics Research* 35.*14* (2016), pp. 1750–1759, DOI: 10.1177/0278364916672311.

[145] Trevor Standley, « Finding Optimal Solutions to Cooperative Pathfinding Problems », *in*: *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, Atlanta, Georgia, 2010, pp. 173–178.

[146] Athony Stentz, « Optimal and Efficient Path Planning for Unknown and Dynamic Environments », *in*: *IJRA* 10 (Feb. 1993).

[147] L. J. Stockmeyer and A. R. Meyer, « Word Problems Requiring Exponential Time (Preliminary Report) », *in*: *Proceedings of the Fifth Annual ACM Symposium on Theory of Computing (STOC)*, 1973, pp. 1–9, ISBN: 9781450374309, DOI: 10.1145/800125.804029.

[148] Bryan Stout, *Smart Move: Intelligent Path-Finding*, 1996.

[149] Nathan R. Sturtevant, « Benchmarks for Grid-Based Pathfinding », *in*: *IEEE Trans. Comput. Intellig. and AI in Games* 4.*2* (2012), pp. 144–148, DOI: 10.1109/TCIAIG.2012.2197681, URL: https://doi.org/10.1109/TCIAIG.2012.2197681.

[150] Pavel Surynek, « An Optimization Variant of Multi-Robot Path Planning is Intractable », *in*: *Proc. of AAAI*, 2010, pp. 1261–1263.

[151] D. Tateo, J. Banfi, A. Riva, F. Amigoni, and A. Bonarini, « Multiagent Connected Path Planning: PSPACE-Completeness and How to Deal With It », *in*: *AAAI 20018,* 2018, pp. 4735–4742.

[152] WT Luke Teacy, Jing Nie, Sally McClean, and Gerard Parr, « Maintaining connectivity in UAV swarm sensing », *in*: *2010 IEEE Globecom Workshops*, IEEE, 2010, pp. 1771–1776.

[153] Sebastian Thrun, « Robotic Mapping: A Survey », *in*: *Exploring Artificial Intelligence in the New Millennium*, 2003, pp. 1–35.

[154] T. Thurston and Huosheng Hu, « Distributed agent architecture for port automation », *in*: *Proceedings 26th Annual International Computer Software and Applications*, 2002, pp. 81–87, DOI: 10.1109/CMPSAC.2002.1044537.

[155] Hudson Turner, « Polynomial-Length Planning Spans the Polynomial Hierarchy », *in*: *Logics in Artificial Intelligence, European Conference, JELIA 2002, Cosenza, Italy, September, 23-26, Proceedings*, 2002, DOI: 10.1007/3-540-45757-7_10.

[156] A. Ulusoy, S. L. Smith, X. C. Ding, and C. Belta, « Robust multi-robot optimal path planning with temporal logic constraints », *in*: *Proc. of ICRA*, 2012, pp. 4693–4698, DOI: 10.1109/ICRA.2012.6224792.

[157] Manuela Veloso, Joydeep Biswas, Brian Coltin, and Stephanie Rosenthal, « CoBots: Robust Symbiotic Autonomous Mobile Service Robots », *in*: *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, Buenos Aires, Argentina, 2015, pp. 4423–4429, ISBN: 9781577357384.

[158] Glenn Wagner and Howie Choset, « M*: A complete multirobot path planning algorithm with performance bounds », *in*: *International Conference on Intelligent Robots and Systems*, 2011, pp. 3260–3267, DOI: 10.1109/IROS.2011.6095022.

[159] Ko-Hsin Cindy Wang, Adi Botea, et al., « Fast and Memory-Efficient Multi-Agent Pathfinding. », *in*: *ICAPS*, 2008, pp. 380–387.

[160] Ko-Hsin Cindy Wang and Adi Botea, « Tractable Multi-Agent Path Planning on Grid Maps », *in*: *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 2009, pp. 1870–1875.

[161] Kai M Wurm, Cyrill Stachniss, and Wolfram Burgard, « Coordinated multi-robot exploration using a segmentation of the environment », *in*: *IROS*, IEEE, 2008, pp. 1160–1165.

[162] Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz, « Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses », *in*: *National Conference on Innovative Applications of Artificial Intelligence*, IAAI'07, Vancouver, British Columbia, Canada, 2007, pp. 1752–1759, ISBN: 9781577353232.

[163] Xiaoming Zheng, Sonal Jain, S. Koenig, and D. Kempe, « Multi-robot forest coverage », *in*: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 3852–3857, DOI: 10.1109/IROS.2005.1545323.

[164] Ling Xu, « Graph Planning for Environmental Coverage », PhD thesis, Carnegie Mellon University, Aug. 2011.

[165] Yinfeng Xu, Maolin Hu, Bing Su, Binhai Zhu, and Zhijun Zhu, « The canadian traveller problem and its competitive analysis », *in*: *J. Comb. Optim.* 18 (Aug. 2009), pp. 195–205, DOI: 10.1007/s10878-008-9156-y.

[166] E. Yanmaz, « Connectivity versus area coverage in unmanned aerial vehicle networks », *in*: *Proceedings of IEEE International Conference on Communications, ICC 2012,* 2012, DOI: 10.1109/ICC.2012.6364585.

[167] Jingjin Yu, « Intractability of Optimal Multirobot Path Planning on Planar Graphs », *in*: *IEEE Robotics and Automation Letters* 1.*1* (2016), pp. 33–40, DOI: 10.1109/LRA.2015.2503143.

[168] Jingjin Yu and Steven LaValle, « Planning Optimal Paths for Multiple Robots on Graphs », *in*: *Proceedings - IEEE International Conference on Robotics and Automation* (2012), pp. 3612–3617, DOI: 10.1109/ICRA.2013.6631084.

[169] Jingjin Yu and Steven M. LaValle, « Multi-agent Path Planning and Network Flow », *in*: *Algorithmic Foundations of Robotics X*, Berlin, Heidelberg, 2013, pp. 157–173, ISBN: 978-3-642-36279-8.

[170] Jingjin Yu and Steven M. LaValle, « Optimal Multirobot Path Planning on Graphs: Complete Algorithms and Effective Heuristics », *in*: *IEEE Trans. on Rob.* 32.*5* (2016), pp. 1163–1177, DOI: 10.1109/TRO.2016.2593448.

[171] Jingjin Yu and Steven M. LaValle, « Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs », *in*: *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, AAAI'13, Bellevue, Washington, 2013, pp. 1443–1449.

[172] Jingjin Yu and Daniela Rus, « Pebble Motion on Graphs with Rotations: Efficient Feasibility Tests and Planning Algorithms », *in*: *Algorithmic Foundations of Robotics*, vol. 107, Springer Tracts in Advanced Robotics, 2014, pp. 729–746, DOI: 10.1007/978-3-319-16595-0_42.

[173] Yun Seok Nam, Bum Hee Lee, and Nak Yong Ko, « A View-Time Based Potential Field Method for Moving Obstacle Avoidance », *in*: *Proc. of SICE*, 1995, pp. 1463–1468, DOI: 10.1109/SICE.1995.526730.

[174] Huili Zhang and Yinfeng Xu, « The k-Canadian Travelers Problem with Communication », *in*: *FAW-AAIM*, 2011, pp. 17–28, DOI: 10.1007/s10878-012-9503-x.

INDEX OF DEFINITIONS

DOCTORAT BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE RENNES 1

Titre : Recherche de Chemin Multi-agents Connectés : Comment les Robots Téléphonent au Volant en Toute Impunité

Mot clés : Plannification, Système Multi-Agents, Théorie de la complexité, Connectivité, Connaissance Incomplète

Résumé : La planification de chemin consiste à concevoir une séquence d'étapes à suivre pour une entité mobile. Cette tâche est au cœur de nombreux problèmes du monde réel. L'étude de la planification autonome peut permettre de réduire la congestion, la pollution, les accidents, les coûts et plus encore. Dans certaines applications, il est important de considérer la connectivité des agents. Bien que certaines configurations garantissent une connectivité permanente entre les entités, ce n'est pas toujours le cas dans les applications avec des environnements ouverts. On retrouve aussi, dans de nombreuses applications, le manque de connaissance complète de la zone dans laquelle les entités se déplacent. Par exemple, dans les missions d'exploration, les agents ne reçoivent aucune information sur l'environnement et doivent le découvrir par eux-mêmes. Un problème important, appelé Multi-Agent Path Finding, consiste à trouver une séquence d'étapes pour qu'un groupe d'agents atteigne des cibles spécifiées tout en évitant les collisions. Tout d'abord, nous présentons un cadre pour étudier et modéliser les problèmes de planification de chemins multi-agents basés sur la connectivité. Nous fournissons un travail initial détaillé sur la complexité de ce cadre et un algorithme optimal pour le résoudre. Deuxièmement, nous étendons notre cadre de connectivité au cadre de connaissance incomplète et montrons la complexité du calcul connecté et décentralisé des plans dans des environnements partiellement connus.

Title: Connected Multi-Agent Path Finding: How Robots Get Away with Texting and Driving

Keywords: Planning, Multi-Agent Systems, Complexity Theory, Connectivity, Incomplete Knowledge

Abstract: Path planning is the task of devising a sequence steps for a mobile entity to follow. This task is required at the center of numerous real-world problems. The study of autonomous planning can allow one to reduce congestion, pollution, accidents, costs and more. In some applications, it is important to consider the connectivity of the agents. Although some settings guarantee a permanent connectivity among entities, this is not always true in applications with open environments. Another aspect that can be found in many applications is the lack of complete knowledge of the area in which the entities move. For instance, in exploration missions, the agents are not provided any information of the environment and must discover it by themselves. An important problem, called Multi-Agent Path Finding, is to find a sequence of steps for a group of agents to reach specified targets while avoiding collisions. First, we present a framework to study and model connectivity-based multi-agent path planning problems. We provide a detailed initial work on the complexity of this framework and an optimal algorithm to solve it. Second, we extend our connectivity framework to the incomplete knowledge setting and show the complexity of the connected and decentralized computation of plans under partially known environments.