PARAMETERIZED COMPLEXITY OF COORDINATED MOTION PLANNING

A THESIS

submitted in partial fulfillment of the requirements
for the award of the dual degree of

Bachelor of Science-Master of Science

in

ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

by

FATHIMA HENNA

(21105)

DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
INDIAN INSTITUTE OF SCIENCE EDUCATION AND RESEARCH BHOPAL
BHOPAL - 462 066

April 2026

भारतीय विज्ञान शिक्षा एवं अनुसंधान संस्थान भोपाल
Indian Institute of Science Education and Research Bhopal
Department of Electrical Engineering and Computer Science
Bhopal By-pass Road, Bhauri, Bhopal 462066

CERTIFICATE

This is to certify that Fathima Henna, BS-MS (Dual Degree) student in Department of Electrical Engineering and Computer Science, has completed bona fide work on the thesis entitled **'Parameterized Complexity of Coordinated Motion Planning'** under my supervision and guidance.

April 2026 **Dr. P. B. Sujit** **Dr. Prafullkumar Tale**
IISER Bhopal **(IISER Pune)**

Committee Member	**Signature**	**Date**
Dr. P. B. Sujit	_____	_____
Dr. Ankur Raina	_____	_____
Dr. Arijit Sen	_____	_____
Dr. Priyadarshi Mukherjee	_____	_____
Dr. Santanu Talukder	_____	_____

ACADEMIC INTEGRITY AND COPYRIGHT DISCLAIMER

I hereby declare that this project report is my own work and due acknowledgement has been made wherever the work described is based on the findings of other investigators. This report has not been accepted for the award of any other degree or diploma at IISER Bhopal or any other educational institution. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

I certify that all copyrighted material incorporated into this document is in compliance with the Indian Copyright (Amendment) Act (2012) and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and safeguard IISER Bhopal from any claims that may arise from any copyright violation.

April 2026 **Fathima Henna**
IISER Bhopal

ACKNOWLEDGEMENT

First and foremost, I would like to express my sincere gratitude towards my advisor Dr. Prafullkumar Tale for his support and guidance throughout my master's thesis. As my first research mentor, his constant efforts to encourage, reassure, and instill foundational research skills in me were very helpful and what I needed as a novice.

I am thankful to my co-supervisor, Dr. P. B. Sujit, for his guidance and thoughtful suggestions. Sincere gratitude to Dr. Pradeesha Ashok for her collaboration and meticulous inputs towards our work, which offered new insights that improved the final outcome of this thesis.

I would not have had an opportunity to do this thesis if not for my parents, who wanted to give me the best possible education. Thank you for being ambitious and hopeful for me. To my siblings, thank you for reminding me of the feeling of home whenever we spoke on the phone.

My sincere thanks to my peers Rohith and Priyanshu for being there to discuss the day-to-day happenings of my research, which helped me stay zestful whenever I was starting to feel a little dejected. Thank you for your thoughtful feedback and suggestions.

Thanks to my friend Aiswarya for listening to me vent about how stressful research is whenever I was stuck, and for letting me know how stressed you were about your own work. Thanks to Navaneetha and Daya for all our conversations on our life conundrums. Of course, I have to thank Anandu for his constant unsolicited advice and reality checks. Thanks to JV, KP, and all my friends for all the fun times we had.

Most importantly, thanks to everyone who, during the course of this thesis, gave me suggestions, showed me kindness, and spoke to me about my work. Words cannot express how grateful I am for all of you; I truly appreciate how you made this journey better.

Fathima Henna

ABSTRACT

While traditional Multi-Agent Path Finding (MAPF) focuses on routing agents to fixed targets, real-world environments like warehouses and parking lots often contain movable, unassigned obstacles that significantly complicate coordination. This thesis explores the computational complexity of Generalized Coordinated Motion Planning (GCMP) on graphs, which incorporates these "obstacle robots", both labeled and unlabeled variants, under total length and makespan optimization objectives respectively. The research investigates whether the inherent tractability of unlabeled robot motion persists when unassigned obstacles are introduced.

For the labeled variant LGCMP-ℓ, the study presents an algorithm with a time complexity of $n^{O(k+m)}$, placing the problem in the complexity class **XP** when parameterized by the total number of worker robots (k) and obstacle robots (m). This result establishes that the problem is solvable in polynomial time for a fixed number of robots. In the unlabeled variant UGCMP-μ, the thesis proves that the problem becomes **NP-hard** even for a small makespan of $\mu = 4$. This finding demonstrates that robot interchangeability cannot mitigate the combinatorial complexity introduced by obstacles. Conversely, the study provides a polynomial-time algorithm for the case of $\mu = 1$ using a reduction to the single-commodity maximum flow problem. Finally, structural analysis of distance optimization on tree graphs with one obstacle shows that the obstacle's movement is effected by the balance of sources and sinks in connected subtrees.

LIST OF ABBREVIATIONS

CMP	Coordinated Motion Planning
GCMP	Generalized Coordinated Motion Planning
LGCMP	Labeled GCMP
LGCMP-ℓ	LGCMP with Length Minimization
UGCMP	Unlabeled GCMP
UGCMP-ℓ	UGCMP with Length Minimization
UGCMP-μ	UGCMP with Makespan Minimization
3DM	3-Dimensional Matching

LIST OF SYMBOLS

Input Instance

a_i	Start vertex of worker robot r_i
b_i	Target vertex of worker robot r_i
C_t	System configuration at time step t
$G(V, E)$	The input graph with vertex set V and edge set E
$G_T(V, E)$	The input graph when restricted to a tree structure
k	Number of worker robots
$G_T[\mathcal{K}_i]$	The i-th connected component of a rooted tree G_T
ℓ	Total length; cumulative edge traversals over a schedule
m	Number of obstacle robots
n	Number of vertices $\lvert V(G)\rvert$
o_i	i-th obstacle robot
$\mathcal{P}$	Start–target pairs $\{(a_i, b_i)\}$ of labeled worker robots
r_i	i-th worker robot
S	Start vertices of unlabeled worker robots
T	Target vertices of unlabeled worker robots
μ	Makespan; number of time steps until all targets occupied
Ω	Start vertices of obstacle robots
τ_{av}	Average arrival time across all worker robots
τ_i	Minimum arrival time of r_i at its target

Graph Notation

$\mathrm{dist}(u, v)$	Shortest path length between u and v
$E(G)$	Edge set of a graph G
$N(v)$	Neighbors of vertex v
$V(G)$	Vertex set of a graph G
$w(u, v)$	Weight of edge (u, v)
Δ	Maximum degree

Complexity Notation

FPT	Complexity class: Fixed-Parameter Tractable
NP	Complexity class: Nondeterministic Polynomial time
$\mathcal{O}(\cdot)$	Big O notation (asymptotic upper bound)
P	Complexity class: Polynomial time
XP	Complexity class: Slice-wise Polynomial

Arena Graph

$\mathcal{A}$	Arena graph of a GCMP instance
E_A	Edges; each represents a valid one-step transition
$\mathcal{S}_A$	Initial configuration vertex
$\mathcal{T}_A$	Set of valid final configuration vertices
V_A	Vertices; each represents a valid system configuration

Flow Network

E'	Self-loop edge set
f_H^*	Maximum flow value in H
H	Flow network for UGCMP-μ with $\mu = 1$
P_1, P_2, P_2', P_3	Vertex partitions in H
α	Super-source of H
β	Super-sink of H

3-Dimensional Matching Reduction

A, B, C	Disjoint sets of size q in 3DM instance
I	3DM instance
I'	Reduced UGCMP-μ instance
M	Set of triples $M \subseteq A \times B \times C$
M'	Exact matching; subset of q triples covering $A \cup B \cup C$
m_p	p-th triple $(a_{i_p}, b_{j_p}, c_{l_p}) \in M$
q	Size of each set A, B, C
$\Omega_{M'}$	Obstacle vertices corresponding to triples in M'

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

Contents

Chapter 1

Introduction

In multi-robot systems, Coordinated motion planning (CMP) (also called Multi-agent Path Finding (MAPF)) is an important problem that involves determining feasible and efficient paths for multiple distinct robots operating in a shared environment. The aim is to move a set of robots from their initial positions to target positions while avoiding collisions and minimizing the total length traversed or the time taken. A well-studied problem in the fields of theory of computation, computational geometry, robotics, and artificial intelligence, its applications range from directing the motion of autonomous robots on networks of narrow tracks to digital games and puzzles.

In continuous, real-world environments, this problem is known to be PSPACE-hard [1]. As a result, it is interesting to look at the case where the environment is abstracted to a discrete graph setting, in which the problem remains NP-hard [2, 3, 4]. In this thesis, we study Generalized coordinated motion planning (GCMP) on graphs, which incorporates additional obstacle robots with no target vertices into the CMP formulation. We explore GCMP under both the classical setting with distinct robots and a variant where the robots are indistinguishable.

1.1 Background and Motivation

When problems are abstracted to graphs, the geometry in real-world instances is replaced by vertices and their adjacencies. In CMP, a robot can move from one vertex to another if the two are connected by an edge, and the weight of the edge denotes the cost of this movement. This helps us study the hardness of the problem purely due to combinatorial factors. The results remain directly applicable to uniform geometric settings with identical robots.

CMP is a generalization of the famous n^2-1 PUZZLE, where n^2-1 distinct robots are placed on the nodes of an $n \times n$ grid graph and the goal is to find a schedule that takes the robots to their desired final positions and minimizes the total length traversed by the robots. This problem is NP-complete [5]. An early generalization of $n^2 - 1$ PUZZLE called COORDINATED PEBBLE MOTIONS ON SIMPLE GRAPHS considers distinct pebbles on a general graph where pebbles can be moved one at a time, and the goal is to find a solution if feasible, that may or may not be optimal. This problem is solvable in polynomial time and the total number of moves needed is tightly bounded by $\times(|V|^3)$, where V is the set of vertices in the input graph [6]. Moreover, testing whether a given instance is feasible can be done in linear time [7]. However, when the goal shifts to optimization, generalizing the n^2-1 PUZZLE to finding optimal routing sequences on arbitrary graphs with any number of robots is known to be NP-hard [2].

With advances in multi-robot systems, where robots can communicate and coordinate, attention shifted to variants in which multiple robots are allowed to move in the same time step as long as collisions are avoided. The CMP framework explicitly incorporates such coordinated motions. The algorithms dealing with feasibility of the problem were generalized to CMP [8]. When it comes to optimization, finding solutions minimizing either makespan or total length traversed in CMP is NP-hard [3, 4].

Real-world environments like warehouses and parking lots often contain movable, unassigned obstacles. Problems involving such obstacle robots along with the robots with assigned targets has been studied previously

[9, 10, 11]. GCMP incorporates these obstacle robots. Since CMP is simply a case of GCMP where the number of obstacle robots is set to zero and is known to be NP-hard, it immediately follows that finding optimal routing schedules in GCMP is NP-hard.

Given this computational intractability, it is important to look at the problem with additional nuance. Since exact polynomial-time algorithms for optimal routing are highly unlikely to exist, the problem must be approached through alternative algorithmic paradigms. In practical applications, researchers rely on heuristics and approximation algorithms to find sufficiently good, collision-free paths quickly. Another way to approach these intractable problems is through the lens of parameterized complexity. By analyzing the problem against specific parameters such as the number of agents, the underlying graph structure etc., we can isolate the sources of computational hardness and identify conditions under which the problem becomes tractable.

Parallel to the development of routing distinct agents, significant attention has been given to the unlabeled setting, where the robots are indistinguishable and the goal is to ensure that every target destination is eventually occupied by some robot. This variant is highly tractable and optimizing for either total length or makespan can be solved efficiently in polynomial time [12].

In existing formulations with obstacle robots, either the number of target-assigned robot is one [9], or the robots with assigned targets are distinct and labeled. It is compelling to see whether adding obstacles to the polynomial-time solvable CMP with unlabeled robots makes the problem hard.

1.2 Problem Variants

In this section, we briefly outline the various problem variants and their nomenclature used throughout this thesis. A formal definition will be established in chapter 2.

The elementary problem of interest is CMP, which operates on a general graph where the total number of robots is strictly equal to the number of

target vertices. Every robot stars from an unique *source vertex* and must eventually occupy a *target vertex*. Such robots are called *worker robots*. The robot movements are discrete in the sense at any given time step, they can either stay on their current vertex, or move to an adjacent vertex that is free at the end of that time step, through the edge that connects the start and end vertex. The robots must move in a way that will avoid collision, thus they cannot occupy the same vertex or edge at the same time.

We extend this setting by studying Generalized coordinated motion planning (GCMP), which introduces movable robots with no assigned target vertex. We call them *obstacle robots*. In GCMP, the total number of robots exceeds the number of target vertices. The only requirement for an obstacle robot is to move out of the way to clear paths for the worker robots, ultimately resting on any unoccupied vertex once the routing is complete. An illustrative example of a GCMP instance, highlighting the worker robots, obstacle robots, and target vertices, is shown in Figure 1.1.

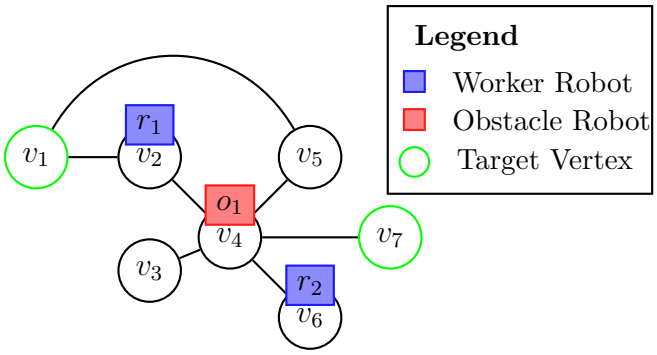


Figure 1.1: An example of a GCMP instance on a 7-vertex graph with two worker robots (r_1, r_2), one obstacle robot (r_3), and two target vertices (v_1, v_7). In UGCMP, the worker robots may occupy either of the two target vertices. In LGCMP, each worker robot must be routed to a specific, pre-assigned target.

Within the GCMP framework, we differentiate the problem based on the distinguishability of the target-assigned robots:

- **Labeled GCMP (LGCMP):** The classical setting where the target-assigned robots are distinct. Each worker robot has a specific, unique

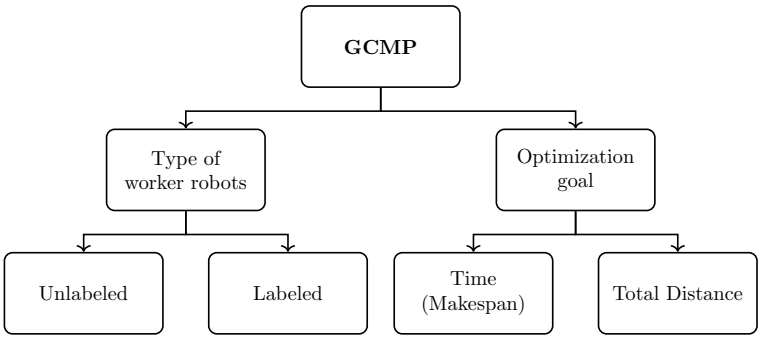


Figure 1.2: Classification of the GCMP problem based on the type of worker robots and the optimization goal.

target vertex that it must reach.

- **Unlabeled GCMP (UGCMP):** A setting where the worker robots are indistinguishable from one another. Any worker robot can occupy any of the available target vertices, provided that all targets are eventually filled.

The aim of these problems is not only to route these robots such that worker robots eventually occupy target vertices, but also to optimize their movements. We analyze the computational complexity of these variants under two primary optimization objectives:

- **Distance Optimization (Total length):** The goal is to minimize the total length traversed, defined as the sum of all individual edge traversals made by all robots (both target-assigned and obstacles) throughout the entire schedule. In physical systems, this directly correlates to the total energy spent.

- **Time Optimization (Makespan):** The goal is to minimize the total time taken for the worker robots to reach targets. Since multiple robots are permitted to move simultaneously during a single time step, the makespan represents the total number of parallel steps required until all target vertices are occupied.

A summary of this classification is shown in Figure 1.2. In this thesis, we study two variants, Labeled GCMP with Length Minimization

(LGCMP-ℓ) and Unlabeled GCMP with Makespan Minimization (UGCMP-μ). Additionally, make some observations on Tree-UGCMP-ℓ with One Obstacle Robot problem.

1.3 Related Work

In recent years, multiple studies have looked into the parameterized complexity of robot motion planning [10, 13, 14]. Eiben et al. [13] study the parameterized complexity of Labeled CMP on grid graphs, optimizing either the makespan μ or total length traversed ℓ. They establish that Labeled CMP on grids is Fixed-Parameter Tractable (FPT) when parameterized by the number of robots k for both optimization goals. However, when parameterized by the optimization variable, the total-length optimization variant is FPT, whereas the makespan optimization variant is not.

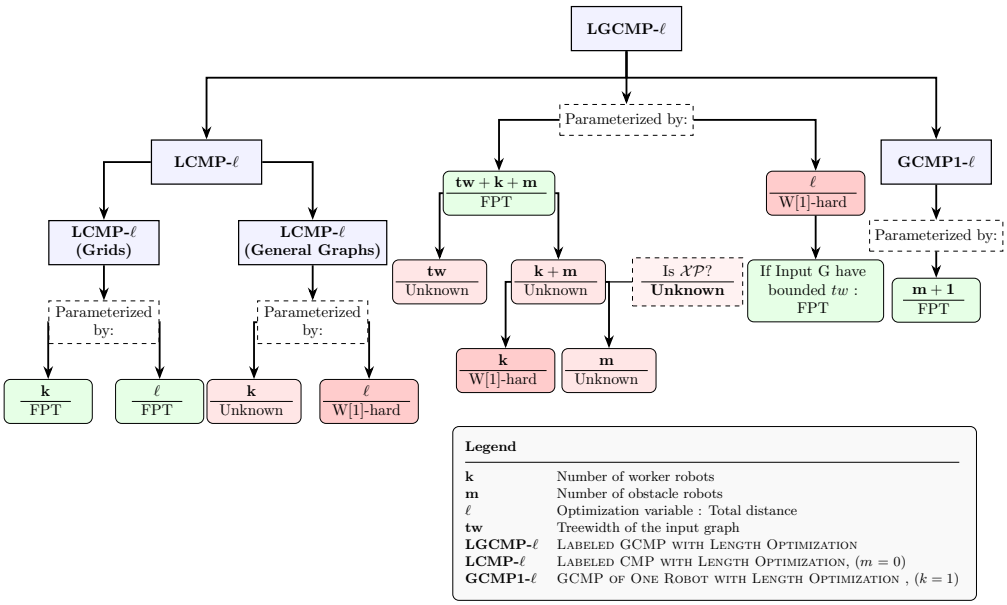


Figure 1.3: Complexity relationships of Labeled GCMP with Length Minimization (LGCMP-ℓ) and its subproblems.

In subsequent work, these results are extended to LGCMP, where the input graph is not restricted to grids, obstacle robots are introduced, and

the aim is to minimize total length traversed [10]. They design an FPT approximation algorithm for LGCMP with an additive error of $\mathcal{O}((k+m)^5)$, where k is the number of worker robots and m is the number of obstacle robots. They further establish that a variant with a single worker robot is FPT parameterized by the total number of robots $m+1$. This result expanded the existing knowledge on the problem which was known to be NP-complete when parallel movements are not allowed [9]. They also found that LGCMP is FPT parameterized by $k+m+tw$, where tw denotes the treewidth of the input graph. However, they find that LGCMP is W[1]-hard when parameterized by ℓ. Thus, the results established on solid grids do not generalize to general graphs. For a finer analysis, they show that Labeled CMP is FPT parameterized by ℓ on graphs with bounded local treewidth, increasing the range of input graphs for which the tractability results apply.

Figure 1.3 summarizes relevant results related to length optimization of CMP and shows the open questions in parameterized complexity of length optimization of GCMP and its subproblems. Notice that while LGCMP-ℓ is FPT parameterized by number of robots and treewidth, it is w[1]-hard when parameterized by the number of worker robots alone. The parameterized complexity of LGCMP-ℓ parameterized by the total number of robots $k+m$ is an interesting problem that remains open. In this thesis, we aim to make progress in regards to this parameter.

On the Unlabeled front, Yu and LaValle demonstrated that when worker robot labels are dropped, the problem drastically simplifies [12]. By reducing the routing problem to a maximum network flow problem on a time-expanded graph, they proved that unlabeled CMP can be solved optimally in polynomial time for both makespan (μ) and total length (ℓ) objectives. Thus, while routing distinct robots is NP-hard, routing indistinguishable and interchangeable robots is highly tractable.

To close the gap between the highly tractable unlabeled variant and the intractable labeled variant of CMP, Colored MAPF problems in which worker robots are separated into teams were studied. Each team of robots are assigned to a set of target vertices equal to the number of robots in the team, and any robot in the team can occupy any of these target vertices. The target

vertex set for a team is disjoint from the target vertex set of another team. In contrast to the single team case , the problem becomes fundamentally harder as more teams are introduced. Yu and LaValle established that finding a time-optimal schedule remains NP-hard even when there are only two distinct groups of interchangeable robots [4].

Most of the research in the unlabeled or colored cases focuses on CMP. Not much is known about changes the addition of obstacle robots can cause to these bounds. It is an open question whether the polynomial-time guarantees of Yu and LaValle for length or makespan optimization hold when obstacle robots are introduced. We addresses this gap by looking into the tractability of UGCMP-μ.

1.4 Contributions

In this thesis, we study the computational complexity of GCMP focusing on the impact of obstacle robots for different optimization parameters. We prove that LGCMP-ℓ belongs to the complexity class XP when parameterized by the total number of robots $k + m$ (Theorem 3.1). We establish UGCMP-μ is NP-hard for a makespan of $\mu = 4$ (Theorem 4.1) and average arrival time $\tau_{av} = 4$ (Corollary 4.1.1), but remains solvable in polynomial time for $\mu = 1$ (Theorem 4.2). This shows that introducing obstacle robots makes the unlabeled variant strictly harder for the makespan objective. We also provide a preliminary case analysis for length minimization on tree graphs with a single obstacle robot.

Chapter 2

Preliminaries

In this chapter, we establish the formal notations and theoretical frameworks that lay the foundation for the remainder of this thesis. We begin by defining basic notations related to graphs that will be used throughout the thesis and briefly introduce standard graph algorithms for finding shortest path on a weighted graph and maximum flow on a network. Then we outline the concepts of classical computational complexity and parameterized complexity, followed by defining polynomial time reductions and parameterized reductions as techniques to transfer hardness between problems. Finally, we establish the formal problem definitions for the variants of GCMP we discuss in this thesis.

2.1 Graph Theory Preliminaries

2.1.1 Basic Notations

A standard graph is denoted as $G = (V, E)$, where V represents a finite set of vertices and $E \subseteq V \times V$ represents the set of edges connecting these vertices. Throughout this thesis, when a graph is provided as part of a problem instance, we will refer to it simply as G, and V and E will be used to denote its vertex and edge sets, respectively. We denote the total number of vertices $|V|$ as n. To prevent any notational ambiguity with the variable m, which is strictly reserved in this work to denote the number of obstacle

robots, the total number of edges will be denoted as $|E|$. We say a graph is simple if it does not have any parallel edges or loops.

The set of all neighbors of a vertex v is denoted by $N(v)$. In weighted graphs, a weight function $w : E \to \mathbb{R}^+$ assigns a numerical value to each edge. We use $dist_{(u,v)}$ to denote the length of the shortest path between u and v.

2.1.2 Shortest Path and Dijkstra's Algorithm

In the shortest path problem, the aim is to find a path between two vertices in a graph such that the sum of the weights of the edges is minimized. For graphs with non-negative edge weights, Dijkstra's algorithm is an efficient method for solving the shortest path problem [15]. The graph is traversed while maintaining a priority queue of vertices keyed by their tentative distance from the source, iteratively recalculating these values during traversal to find the optimal path. Using binary heaps to implement priority key, the time complexity of Dijkstra's algorithm is $\mathcal{O}((|V| + |E|)log|V|)$.

2.1.3 Maximum Flow and Edmonds-Karp Algorithm

A flow network is a directed graph $H = (V, E, c, s, t)$ where each edge $(u, v) \in E$ has a non-negative capacity $c(u, v)$, $s \in V$ is the source vertex, and $t \in V$ is the sink vertex. The single commodity maximum flow problem is to route as much flow as possible from s to t without exceeding the capacity of any edge, subject to flow conservation at all intermediate nodes.

The Edmonds-Karp algorithm is an implementation of the Ford-Fulkerson algorithm for computing the maximum flow [15]. It computes paths from s to t with available capacity called 'augmenting paths' using Breadth-First Search (BFS). The Edmonds-Karp algorithm runs in $\mathcal{O}(|V||E|^2)$.

2.1.4 Arena graphs

An arena graph $\mathcal{A}$ (also referred to as a game graph or state-space graph) is a graph where each node represents a complete, valid state of the system.

An edge exists between vertices $u_{\mathcal{A}}$ and $v_{\mathcal{A}}$ if and only if the system can transition from the states represented by these vertices via a single, valid event. To analyze complex systems like GCMP it is usually beneficial to model the configurations of the entire system and transitions between them in discrete time steps, rather than tracking individual robots within the problem environment.

We define an arena graph as a tuple $\mathcal{A} = (V_{\mathcal{A}}, E_{\mathcal{A}}, \mathcal{S}_{\mathcal{A}}, \mathcal{T}_{\mathcal{A}})$, where:

- $V_{\mathcal{A}}$ is the finite set of vertices, where each vertex represents a unique, valid configuration of the system.
- $E_{\mathcal{A}} \subseteq V_{\mathcal{A}} \times V_{\mathcal{A}}$ is the set of edges. An edge exists between two configurations if and only if the system can legally transition from one configuration to the other via a single discrete event. If all transitions are reversible, all the edges remain undirected.
- $\mathcal{S}_{\mathcal{A}} \subseteq V_{\mathcal{A}}$ is a non-empty subset representing the valid initial configurations of the system.
- $\mathcal{T}_{\mathcal{A}} \subseteq V_{\mathcal{A}}$ is a non-empty subset representing the final configurations that satisfy the problem's objective.

2.2 Computational Complexity

In complexity analysis, we evaluate the efficiency of algorithms in terms of time (and space) requirements. The complexity is represented as a function of input size of the problem. This helps us compare performance of different algorithms irrespective of the platform used to run it, and understand the scalability of an algorithm. We say an algorithm is *efficient* if it runs in polynomial time. We refer to [16] for information on classical complexity and relevant examples on graphs.

A *decision problem* is a computational problem where the answer to any given instance is either 'yes' or 'no'. An *optimization problem* is a problem where the expected output is a value denoting the best solution out of all the feasible solutions. We formulate problems as decision problems involving the

parameter to be optimized because they allow for a mathematically precise definition of efficiency and complexity classes, while remaining just about as hard to solve as their optimization counterparts.

2.2.1 The class P

The complexity class P consists of all decision problems that can be solved in polynomial time on a deterministic Turing machine. That is, a problem is in P if there exist an algorithm that runs in $O(n^c)$ time, where n is the input size and c is a constant.

2.2.2 The class NP

The class NP consists of all decision problems that, given a *certificate*, can be *verified* in polynomial time by an *efficient certifier*. An efficient certifier is an algorithm that takes two input strings; the input instance of the problem, and a certificate, and verifies in polynomial time that the certificate is a valid proof that the given instance is a 'yes' instance. Clearly, P$\subseteq$ NP.

2.2.3 NP-hard and NP-complete problems

A problem B is NP-hard if every problem A in NP can be reduced to it in polynomial time. That is, there exists a polynomial time algorithm that, given an instance of A, outputs an instance of B, such that the instance of A is a yes-instance if and only if the instance of B is a yes-instance.

A problem is NP-complete if it is both NP-hard and in NP. NP-complete problems are the hardest problems in NP. The existence of an efficient algorithm for an NP-complete problem would imply existence of efficient algorithms for all problems in NP, proving P=NP. However, no polynomial-time algorithm is known for any NP-complete problem, and it is widely believed that such algorithms do not exist. Thus, it is conjectured that P$\neq$NP. As a result, discovering that a problem is NP-hard or NP-complete provides strong evidence that no efficient algorithm exists for solving it.

2.3 Polynomial-Time Reductions

Polynomial time reductions are a tool used to compare relative hardness of two problems and is used to prove that a problem is NP-hard. If there exist a polynomial time reduction from A to B, denoted by $A \leq_p B$, then B is *at least as hard as* A. Then, if A is NP-hard, B is also NP-hard.

Definition 2.1. Polynomial-time reduction. Let $A, B \subseteq \Sigma^*$ be two problems. A *polynomial-time reduction* from A to B is a function $f : \Sigma^* \to \Sigma^*$ that, given an instance $A(x)$ outputs an instance $B(x')$ such that:

1. $A(x)$ is a yes-instance if and only if $B(x')$ is a yes-instance, and
2. the time to compute f is a polynomial function of $|x|$.

If such a reduction exists, we write $A \leq_p B$.

2.4 Parameterized Complexity

Classical complexity theory often categorizes NP-hard problems as universally intractable. Parameterized complexity provides a finer framework where the time complexity of a problem is analyzed based on the input size n and an additional parameter k, often representing a structural aspect of the instance or a measure of size of the output. We refer to [17] for details on parameterized complexity.

Definition 2.2. Parameterized problem. A parameterized problem Π is a subset of $\Sigma^* \times \mathbb{N}$, where Σ is a fixed finite alphabet. An instance of Π is denoted as a pair (x, k), where $x \in \Sigma^*$ represents the classical problem input and $k \in \mathbb{N}$ is an integer referred to as the *parameter*.

2.4.1 Fixed Parameter Tractable (FPT) Problems

A parameterized problem Π is said to be fixed parameter tractable if there exists an algorithm that can solve any instance $\Pi(x, k)$ in time $f(k).O(n^c)$, where $n = |x|$, f is a computable function, and c is a constant not dependent

on n or k. Class FPT is the class of all such problems. The combinatorial explosion is restricted to the parameter k, leaving the polynomial dependence on the input size n completely independent of k.

2.4.2 Slice-wise Polynomial (XP) Problems

A parameterized problem Π is said to be **slice-wise polynomial (XP)** if there exists an algorithm that can solve any instance $\Pi(x, k)$ in time $f(k) \cdot n^{g(k)}$ where $n = |x|$, and f and g are computable functions that depends only on k. Class XP is the class of all such problems. Unlike FPT, the parameter k appears in the exponent of the polynomial. While the problem is polynomial-time solvable for any fixed constant k the degree of the polynomial grows as k increases, making it less tractable than FPT problems.

2.4.3 W-Heirarchy and W[1]-Hardness

The W-hierarchy classifies parameterized problems into a hierarchy of the following form:

$$\mathsf{FPT} \subseteq \mathrm{W}[1] \subseteq \mathrm{W}[2] \subseteq \cdots \subseteq \mathrm{W}[t]$$

where $t \geq 1$ is an integer.

A parameterized problem is considered *W[t]-hard* if every problem in the class W[t] can be reduced to it using a parameterized reduction. Assuming the standard conjecture that $\mathsf{FPT} \neq \mathrm{W}[1]$, proving a problem is **W[1]-hard** provides strong mathematical evidence that the problem does not admit a fixed-parameter tractable algorithm. As the index t increases, proving W[t]-hardness establishes a stronger degree of fixed-parameter intractability.

2.5 Parameterized Reductions

A parameterized reduction is a technique analogous to polynomial-time reduction, that is used to compare relative hardness of parameterized problems. If there exist a parameterized reduction from A_Π to B_Π where A_Π and B_Π are two parameterized problems parameterized by k and k' respectively, and

A_Π is not FPT when parameterized by k, then it proves that B_Π is unlikely to be FPT when parameterized by k'.

Definition 2.3. Parameterized Reduction. Let $A_\Pi, B_\Pi \subseteq \Sigma^* \times \mathbb{N}$ be two parameterized problems. A *parameterized reduction* from A_Π to B_Π is a function

$$f : \Sigma^* \times \mathbb{N} \longrightarrow \Sigma^* \times \mathbb{N},$$

mapping an instance (x, k) of A_Π to an instance (x', k') of B_Π, such that:

1. (x, k) is a yes-instance of A_Π if and only if (x', k') is a yes-instance of B_Π,
2. f can be computed in time $f(k) \cdot |x|^{O(1)}$ for some computable function f, and
3. the new parameter $k' \leq g(k)$ for some computable function g depending only on k.

If such a reduction exists, we write $A_\Pi \leq_{\mathsf{FPT}} B_\Pi$.

2.6 Formal Problem Definitions

All problems considered in this thesis are defined on finite, simple, undirected graphs with unit length edges. We consider k worker robots, denoted as $r_1, \ldots, r_k$, and m obstacle robots, denoted as $o_1, \ldots, o_m$.

A *configuration* at time step t is defined as an injective mapping C_t that assigns each of the $k + m$ robots to a distinct vertex in V. The robots move in discrete time steps at unit speed. Between time t and $t + 1$, any given robot r_i may either remain stationary ($C_{t+1}(r_i) = C_t(r_i)$) or traverse an edge ($(C_t(r_i), C_{t+1}(r_i)) \in E$).

A sequence of configurations $(C_0, C_1, \ldots, C_\mu)$ is a *collision-free schedule* if it strictly adheres to the following constraints for all $t \geq 0$:

- **No Vertex Collisions:** For any two distinct robots r_1 and r_2, $C_t(r_1) \neq C_t(r_2)$. Since this constraint is evaluated at the end of each discrete

time step, a robot r_1 is permitted to transition to a vertex currently occupied by another robot r_2 at time t, provided that robot r_2 vacates that vertex during the same time step and no other robot other than r_1 moves there.

- **No Edge Collisions:** For any two distinct robots r_1 and r_2, if $C_t(x) = u$ and $C_{t+1}(r_1) = v$, then it cannot be true that $C_t(r_2) = v$ and $C_{t+1}(r_2) = u$. That is, no two robots may travel across the same edge in opposite directions between two time steps.

In addition to these constraints, both problem variants have the following final configuration requirements subject to which the schedule becomes a *valid collision-free schedule*:

- **Obstacle robots:** In both variants, they originate from initial positions Ω and are permitted to occupy any of the vertices that are not worker robot targets in the final configuration C_μ.
- **Worker robots:**
  - For the **Labeled** variant, each worker robot r_i has a unique identity. They start at a_i and should occupy a specific target vertex b_i at $t \geq \mu$. Thus, the final configuration must meet the condition: $C_\mu(r_i) = b_i$ for all $i \in [k]$.
  - For the **Unlabeled** variant, the worker robots are interchangeable. They start from a set of starting vertices S and must together occupy a set of target vertices T. The schedule must satisfy the terminal condition $\{C_\mu(r_i) \mid i \in [k]\} = T$.

To evaluate the efficiency of such schedules, we define two primary optimization parameters:

- **Total Length (ℓ):** The cumulative number of edge traversals across the entire schedule.

$$\ell = \sum_{t=0}^{\mu-1} |\{x \in \{r_1, \ldots, r_k, o_1, \ldots, o_m\} \mid C_t(x) \neq C_{t+1}(x)\}|$$

- **Makespan (μ):** The total time required for the system to reach its final configuration. For each worker robot r_i, let $\tau_i \geq 0$ be the smallest τ such that r_i reaches its final target vertex and remains there for all subsequent time steps. The makespan is defined as the largest of these these values, $\mu = \max_{1 \leq i \leq k} \tau_i$.

In addition to makespan, another well studied goal in time optimization is to minimize the average of **minimum arrival time** τ of k robots. We call this objective the **average arrival time** τ_{av}.

Using the introduced terminology, we formalize LGCMP-ℓ and UGCMP-μ as follows:

LABELED GCMP WITH LENGTH MINIMIZATION (LGCMP-ℓ)
Input: G, a set $\mathcal{P} = \{(a_i, b_i) \mid i \in [k]\}$ denoting the start and target vertices of k worker robots, a set $\Omega \subseteq V$ representing start vertices of m obstacle robots, integer ℓ.
Question: Does there exist a collision-free schedule such that each worker robot r_i moves from a_i to b_i, and the total length is at most ℓ?

UNLABELED GCMP WITH MAKESPAN MINIMIZATION (UGCMP-μ)
Input: G, sets $S \subseteq V$ and $T \subseteq V$ denoting start and target vertices of k unlabeled worker robots, a set $\Omega \subseteq V$ denoting initial positions of m obstacle robots, and a positive integer μ.
Question: Does there exist a collision-free schedule that moves all worker robots from S to T such that the makespan is at most μ?

Chapter 3

Labeled GCMP with Length Minimization

In this chapter, we look into the computational complexity of the LGCMP problem with respect to the total length minimization objective. Specifically, we study the role of the parameter 'total number of robots' in the complexity of the problem.

3.1 Our Results

We establish that the problem is solvable in polynomial time when the total number of robots $k + m$ is a constant (see Theorem 3.1). This result implies that LGCMP-ℓ belongs to class XP when parameterized by the total number of robots.

3.2 Solving LGCMP-ℓ using Arena Graphs

In this section we map an algorithm to solve LGCMP-ℓ by constructing an arena graph $\mathcal{A}$. Then the input instance $(G, \mathcal{P}, \Omega, \ell)$ is reduced to a shortest path problem on $\mathcal{A}$. A summary of the algorithm is given in Table 3.1. We show the complexity bounds of this approach is $n^{\mathcal{O}(n+m)}$.

Theorem 3.1. *Given an instance $(G, \mathcal{P}, \Omega, \ell)$ of* LGCMP-ℓ*, there exist an algorithm that gives an optimal solution in $n^{\mathcal{O}(k+m)}$ time.*

Proof. We prove the theorem by constructing an arena graph $\mathcal{A}$, and proving that the input LGCMP-ℓ instance reduces to a single-source shortest path problem on $\mathcal{A}$.

Construction of $\mathcal{A}$. Given an instance $(G, \mathcal{P}, \Omega, \ell)$ of LGCMP-ℓ, we construct $\mathcal{A} = (V_\mathcal{A}, E_\mathcal{A}, \mathcal{S}_\mathcal{A}, \mathcal{T}_\mathcal{A})$ as follows:

- $V_\mathcal{A}$ consists of all feasible configurations. For a vertex $v \in V_\mathcal{A}$, we represent the corresponding configuration of the LGCMP-ℓ instance as C_v .
- $E_\mathcal{A}$ contains an edge (u, v) if there exists a collision-free schedule of one time step (C_u, C_v).
- Each edge (u, v) is assigned a weight $w(u, v) \in \{0, 1, \ldots, k + m\}$, representing the total number of robots that change their vertex position between C_u and C_v calculated as:

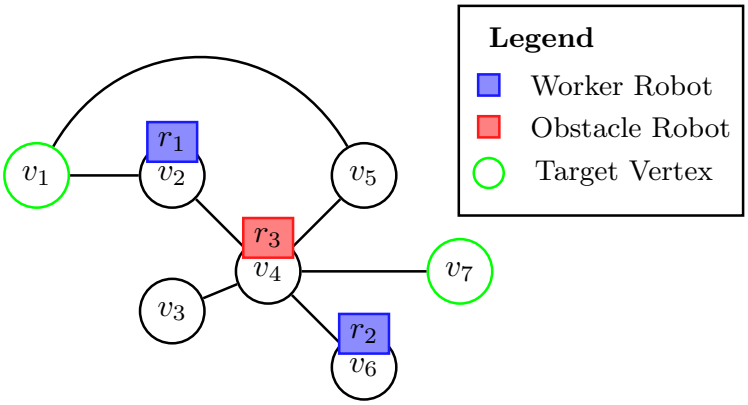
$$w_{(u,v)} = \sum |\{x \in \{r_1, \ldots, r_k, o_1, \ldots, o_m\} \mid C_u(x) \neq C_v(x)\}|$$

- $\mathcal{S}_\mathcal{A} \in V_\mathcal{A}$ corresponds to the unique start configuration of the LGCMP-ℓ instance, where every worker robot r_i is on a_i and obstacle robot o_j is on Ω_j.
- $\mathcal{T}_\mathcal{A} \subseteq V_\mathcal{A}$ corresponds to the set of configurations where every worker robot r_i is at its target b_i.

Figure 3.1 illustrates the arena graph construction for a specific input instance $(G, \mathcal{P}, \Omega, \ell)$.

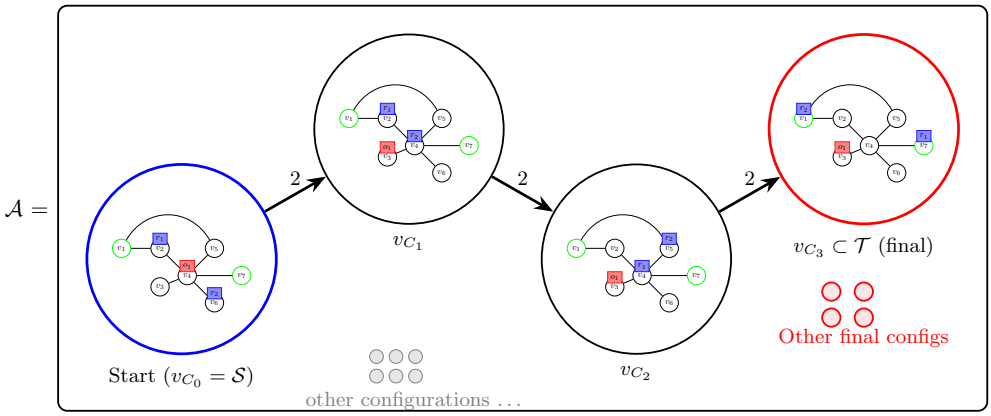
Complexity analysis of arena graph construction. The number of ways to place $k + m$ labeled robots on n vertices is:

$$|V_\mathcal{A}| = \binom{n}{k+m}(k+m)! = n^{\mathcal{O}(k+m)}$$



$$\mathcal{P} = \{(v_2, v_7), (v_6, v_1)\},\ \Omega = \{v_4\}$$

(a) Initial configuration of an input instance $(G, \mathcal{P}, \Omega)$



(b) Arena graph $\mathcal{A}$

Figure 3.1: Visual representation of the arena graph construction. Figure 3.1a shows the start positions and targets on the physical graph G and indicates target vertices of the worker robots, while Figure 3.1b illustrates how the configurations and transitions are mapped to the arena graph $\mathcal{A}$, while showing a valid collision free schedule of 3 time steps that optimizes the total length traversed. Here $(G, \mathcal{P}, \Omega, \ell)$ is a yes instance for $\ell = 6$ units.

Since at each time step any robot can move to it's adjacent vertex or stay in it's current vertex, from any configuration, there are at most $(\Delta+1)^{(k+m)}$ many configuration achievable in one time step , where Δ is the maximum degree of G. Since $\Delta \leq n-1$, the total number of edges becomes:

$$|E_{\mathcal{A}}| \leq |V_{\mathcal{A}}| \cdot n^{k+m} = n^{\mathcal{O}(k+m)}$$

The weight of an edge (u, v) in $\mathcal{A}$ can be computed in $\mathcal{O}(k+m)$ time by comparing the positions of all robots in C_u and C_v. The time to assign weight to all edges in $E_{\mathcal{A}}$ is $\mathcal{O}(|E_{\mathcal{A}}|(k+m)$. Additionally, for each vertex, we check if it belongs to $\mathcal{T}_{\mathcal{A}}$ in $\mathcal{O}(k)$ time by verifying if worker robots $r_1, \cdots, rk$ are at their targets. Thus total time to construct $\mathcal{A}$ is $n^{\mathcal{O}(n+m)}$.

Solving $(G, \mathcal{P}, \Omega, \ell)$ using $\mathcal{A}$. By construction, any path in $\mathcal{A}$ from $\mathcal{S}_{\mathcal{A}}$ to some $u \in \mathcal{T}_{\mathcal{A}}$ corresponds to a valid collision-free schedule and the sum of the weights along this path equals the total length ℓ. Thus, finding the shortest such distance on $\mathcal{A}$, we can find the optimum valid schedule and the minimized total length. We find the minimum distance using Dijkstra's algorithm. An implementation of Dijkstra's algorithm that terminate when shortest distance to any $u \in \mathcal{T}_{\mathcal{A}}$ is shown in Algorithm 1. With a binary heap, the complexity of Dijkstra's algorithm is:

$$O\big((|V_{\mathcal{A}}| + |E_{\mathcal{A}}|) \log |V_{\mathcal{A}}|\big)$$

Substituting $|V_{\mathcal{A}}|, |E_{\mathcal{A}}| = n^{O(k+m)}$, the total running time is $n^{O(k+m)}$. This concludes the proof. □

Algorithm 1: NEAREST_TARGET($\mathcal{A}$)

Input: Arena graph $\mathcal{A} = (V_\mathcal{A}, E_\mathcal{A}, \mathcal{S}_\mathcal{A}, \mathcal{T}_\mathcal{A})$

Output: Target vertex $t \in \mathcal{T}_\mathcal{A}$ closest to $\mathcal{S}_\mathcal{A}$ with minimal distance $\text{dist}(t)$

Initialize:

foreach $u \in V_\mathcal{A}$ **do**

  $\text{dist}(u) \leftarrow \infty$; $\text{prev}(u) \leftarrow \text{nil}$;

$\text{dist}(\mathcal{S}_\mathcal{A}) \leftarrow 0$;

$H \leftarrow \texttt{MakeQueue}(V_\mathcal{A}, \text{dist})$;

while H *is not empty* **do**

  $u \leftarrow \texttt{DeleteMin}(H)$;

  if $u \in \mathcal{T}_\mathcal{A}$ **then**

    Return shortest path from $\mathcal{S}_\mathcal{A}$ to u and $\text{dist}(u)$;

  foreach $(u, v) \in E_\mathcal{A}$ **do**

    if $dist(v) > dist(u) + w(u, v)$ **then**

      $\text{dist}(v) \leftarrow \text{dist}(u) + w(u, v)$;

      $\text{prev}(v) \leftarrow u$;

      $\texttt{DecreaseKey}(H, v)$;

Element	**Description**	**Analysis**
$V_\mathcal{A}$	All configurations of $k + m$ robots on G	$\lvert V\rvert = \binom{n}{k+m}(k+m)!$
$E_\mathcal{A}$	Connect each $u_\mathcal{A}$ to configurations reachable in one time step from C_u	$\lvert E\rvert \leq \lvert V\rvert\, n^{(k+m)}$
$w_{(u,v)}$	Total robot movement from $u \to v$	$w \leq (k + m)$ per edge
Dijkstra's algorithm	Shortest length from start to a final configuration on $\mathcal{A}$	$\mathcal{O}(\lvert V_\mathcal{A}\rvert + \lvert E_\mathcal{A}\rvert \log \lvert V_\mathcal{A}\rvert)$

Table 3.1: Summary of the arena graph bounds and the resulting $n^{O(k+m)}$ algorithm for LGCMP-ℓ.

Chapter 4

Unlabeled GCMP with Makespan Minimization

In this chapter, we look into the computational complexity of UGCMP with respect to the makespan minimization objective.

4.1 Our Results

We establish that the problem UGCMP-μ is generally intractable.

- **General Intractability:** We establish that UGCMP-μ is NP-hard even for a small constant value of $\mu = 4$ (see Theorem 4.1). We also show that UGCMP is NP-hard for the average arrival time objective even when $\tau_{av} = 4$ (see Corollary 4.1.1).
- **A Tractable Case ($\mu = 1$):** We contrast this hardness result by identifying a specific, efficiently solvable case. We prove that the problem admits a polynomial-time algorithm when $\mu = 1$ (see Theorem 4.2). This tractability is a result of the strict structural limitation imposed by $\mu = 1$, the complete absence of backward movement along an edge which simplifies the problem significantly.

4.2 Intractability of UGCMP-μ with $\mu = 4$

Theorem 4.1. UGCMP-μ *is* NP*-hard even for* $\mu = 4$.

Proof. We prove this by reducing in polynomial time from 3-Dimensional Matching (3DM).

3-Dimensional Matching (3DM)
Input: Disjoint sets A, B, and C, each of size q, and a set $M \subseteq A \times B \times C$ of triples.
Question: Does there exist a subset $M' \subseteq M$ of size q such that each element of $A \cup B \cup C$ appears in exactly one triple of M'?

Let $\mathcal{I} = (A, B, C, M)$ be an instance of 3DM, where each triple is denoted by $m_p = (a_{i_p}, b_{j_p}, c_{l_p})$ for $p \in [|M|]$. We construct an instance $\mathcal{I}' = (G, S, \mathcal{T}, \Omega)$ of UGCMP-μ with $\mu = 4$. An illustration of the reduced instance $\mathcal{I}'$ is shown in Figure 4.1.

Tuple gadget. We create a set of vertices $\Omega = \{w_1, w_2, \ldots, w_{|M|}\}$ corresponding to the triples in M, which serve as the start vertices for the obstacle robots.

Set gadgets. For each of the sets A, B, and C, we construct a set gadget which is a collection of *vertex clusters*. Each individual cluster $U = \{v_{(U,1)}, v_{(U,2)}, \ldots, v_{(U,q)}\}$ contains exactly q vertices corresponding to each element of the related set in A, B, and C. We say a collection of clusters $\{U_1, U_2, \ldots, U_n\}$ form a *chain* $U_1 - U_2 - \cdots - U_n$ if they are connected such that vertices of a cluster U_i are adjacent to vertices of the same index in U_{i+1} for each $i \in [q-1]$. For example, in Figure 4.1, $T_3 - T_2 - T_1 - S_1 - S_2 - S_3 - S_4$ forms a chain. We say a cluster is a *bridge cluster* if its vertices are adjacent to vertices in the tuple gadget that correspond to tuples they are part of. For example, in Figure 4.1, S_1 is a bridge cluster. Each set gadget consists of two chains and a bridge cluster in each of these chains. All chains are connected to the tuple gadget only through their respective bridge cluster. For a robot to go from one chain to another, it must pass through the tuple gadget.

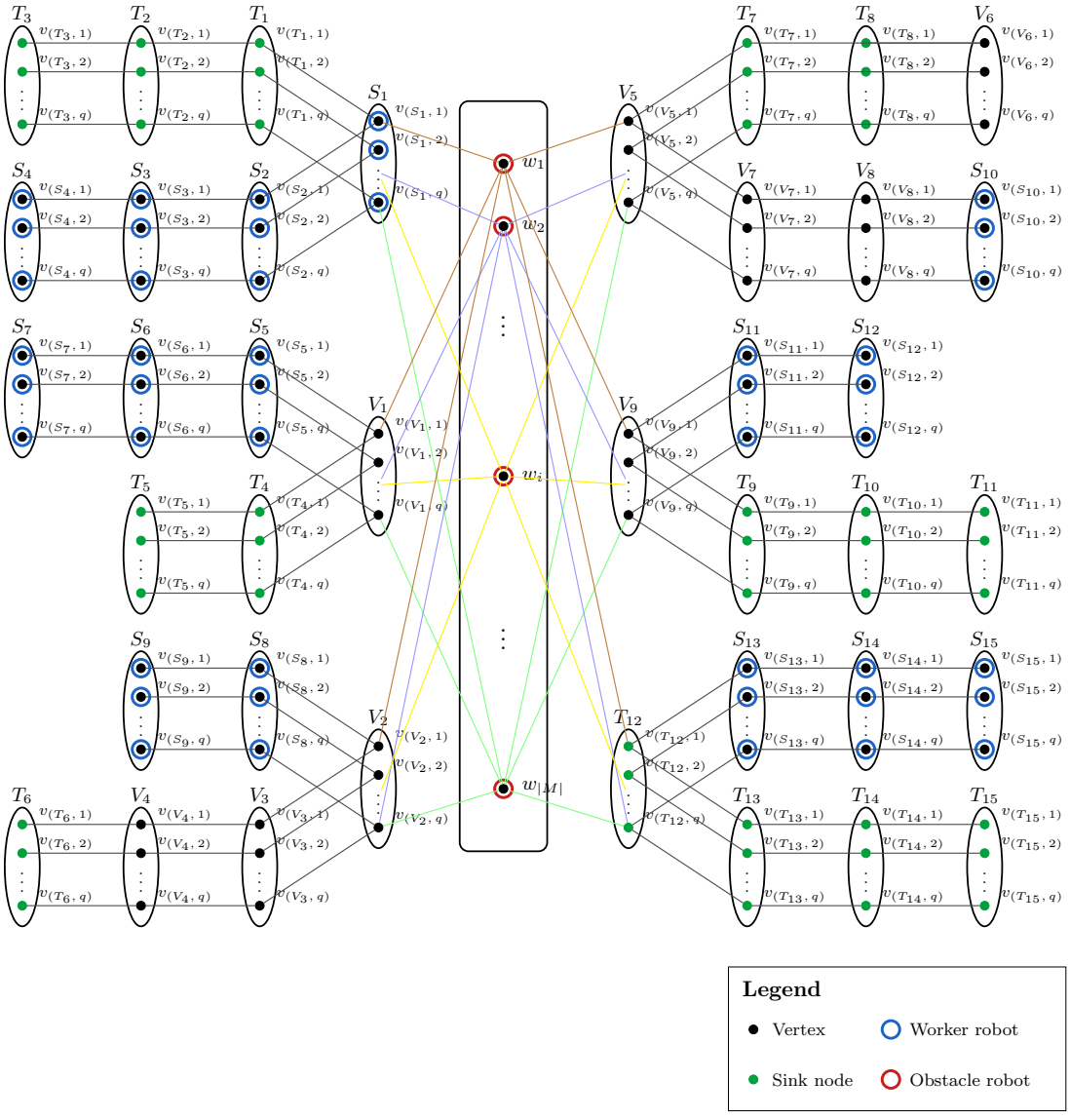


Figure 4.1: An illustration of the reduced Instance $\mathcal{I}'=(G, S, T, \Omega, \mu)$ of UGCMP-μ from $\mathcal{I}$ of 3 Dimensional Matching.

- **A-family gadget:** A-family gadget consists of two chains:
  - Left chain: $T_3 - T_2 - T_1 - S_1 - S_2 - S_3 - S_4$
  - Right chain: $V_6 - T_8 - T_7 - V_5 - V_7 - V_8 - S_{10}$

  We add $S_1 \cup S_2 \cup S_3 \cup S_4 \cup S_{10}$ to the worker robot start vertices S, and $T_1 \cup T_2 \cup T_3 \cup T_7 \cup T_8$ to the target vertices T.

  Among these clusters, S_1 and V_5 are the bridge clusters. For all $i \in [q]$ and $p \in [|M|]$, edges $(v_{(S_1,i)}, w_p)$ and $(v_{(V_5,i)}, w_p)$ exist iff $a_i \in m_p$.

- **B-family gadget:** Similarly, the B-family gadget also consists of two chains:
  - Left chain: $S_7 - S_6 - S_5 - V_1 - T_4 - T_5$
  - Right chain: $S_{12} - S_{11} - V_9 - T_9 - T_{10} - T_{11}$

  We add $S_5 \cup S_6 \cup S_7 \cup S_{11} \cup S_{12}$ to S, and $T_4 \cup T_5 \cup T_9 \cup T_{10} \cup T_{11}$ to T.

  V_1 and V_9 act as bridge clusters. Edges connecting their constituent vertices to $w_p \in \Omega$ exist iff $b_i \in m_p$.

- **C-family gadget:** The C-family gadget consists of:
  - Left chain: $S_9 - S_8 - V_2 - V_3 - V_4 - T_6$
  - Right chain: $S_{15} - S_{14} - S_{13} - T_{12} - T_{13} - T_{14} - T_{15}$

  We add $S_8 \cup S_9 \cup S_{13} \cup S_{14} \cup S_{15}$ to S, and $T_6 \cup T_{12} \cup T_{13} \cup T_{14} \cup T_{15}$ to T.

  V_2 and T_{12} act as bridge clusters. Edges connecting their constituent vertices to $w_p \in \Omega$ exist iff $c_i \in m_p$.

All vertices in these chains that are not in $S \cup \Omega$ are initially empty.

The resulting graph forms the input graph G with $k = 15 \cdot q$ worker robots and $m = |M|$ obstacle robots. This completes the construction of the instance $\mathcal{I}' = (G, S, T, \Omega, \mu)$, where $\mu = 4$.

Complexity of the reduction. The construction of the reduced instance $\mathcal{I}'$ from $\mathcal{I}$ takes polynomial time. Specifically, generating the $|M|$ vertices for the tuple gadget Ω and exactly q vertices for each distinct cluster in the Set gadgets yields a total vertex count of $\mathcal{O}(|M|+q)$. The edges consist of strict one-to-one intra-cluster connections requiring $\mathcal{O}(q)$ edges per chain link, and at most $\mathcal{O}(q \cdot |M|)$ cross-gadget edges between the bridge clusters and Ω. Therefore, the entire graph G and its corresponding subsets S, T, and Ω can be constructed in $\mathcal{O}(q \cdot |M|)$ time and space, which is strictly polynomial with respect to the input size of the 3-Dimensional Matching instance.

Claim 1. *If $\mathcal{I}$ is a YES instance of* 3DM*, then the reduced* UGCMP-μ *instance $\mathcal{I}'$ has a feasible schedule of makespan $\mu \leq 4$.*

Proof of Claim 1. Since $\mathcal{I}$ is a YES instance, there exists $M' \subseteq M$ such that each element of $A \cup B \cup C$ appears in exactly one triple of M'. Let the set of obstacle vertices corresponding to the tuples in M' be denoted as $\Omega_{M'}$. There is a perfect matching between $\Omega_{M'}$ and each of the bridge clusters $S_1, V_1, V_2, V_5, V_9,$ and T_{12}.

We move the robots in the reduced instance $\mathcal{I}'$ such that they follow the target assignments and cluster paths detailed in Table 4.1.

To execute these paths, the schedule proceeds over four time steps as follows:

- $t = 0 \rightarrow 1$: We move the q obstacle robots on $\Omega_{M'}$ to V_5. Simultaneously, worker robots in S_1 advance into the vacated $\Omega_{M'}$. All other worker robots advance one step toward their targets.

- $t = 1 \rightarrow 2$: The obstacle robots continue their path, moving $V_5 \rightarrow T_7$. The worker robots currently in $\Omega_{M'}$ move forward to V_5. This step vacates $\Omega_{M'}$, allowing the worker robots from V_1 to enter. All other workers continue advancing along their assigned paths.

- $t = 2 \rightarrow 3$: The obstacle robots advance toward T_8. The worker robots in $\Omega_{M'}$ move to V_9. With $\Omega_{M'}$ vacated again, worker robots from V_2 now enter $\Omega_{M'}$. All other workers advance one step closer to their targets.

Start Vertex	Target Vertex	Cluster Path
$\Omega_{M'}$	V_6	$\Omega_{M'} \to V_5 \to T_7 \to T_8 \to V_6$
S_1	T_8	$S_1 \to \Omega_{M'} \to V_5 \to T_7 \to T_8$
S_2	T_3	$S_2 \to S_1 \to T_1 \to T_2 \to T_3$
S_3	T_2	$S_3 \to S_2 \to S_1 \to T_1 \to T_2$
S_4	T_1	$S_4 \to S_3 \to S_2 \to S_1 \to T_1$
S_5	T_9	$S_5 \to V_1 \to \Omega_{M'} \to V_9 \to T_9$
S_6	T_5	$S_6 \to S_5 \to V_1 \to T_4 \to T_5$
S_7	T_4	$S_7 \to S_6 \to S_5 \to V_1 \to T_4$
S_8	T_6	$S_8 \to V_2 \to V_3 \to V_4 \to T_6$
S_9	T_{12}	$S_9 \to S_8 \to V_2 \to \Omega_{M'} \to T_{12}$
S_{10}	T_7	$S_{10} \to V_8 \to V_7 \to V_5 \to T_7$
S_{11}	T_{11}	$S_{11} \to V_9 \to T_9 \to T_{10} \to T_{11}$
S_{12}	T_{10}	$S_{12} \to S_{11} \to V_9 \to T_9 \to T_{10}$
S_{13}	T_{15}	$S_{13} \to T_{12} \to T_{13} \to T_{14} \to T_{15}$
S_{14}	T_{14}	$S_{14} \to S_{13} \to T_{12} \to T_{13} \to T_{14}$
S_{15}	T_{13}	$S_{15} \to S_{14} \to S_{13} \to T_{12} \to T_{13}$

Table 4.1: Target cluster assignments for robots and their corresponding routing through clusters.

- $t = 3 \to 4$: All robots take their final step, simultaneously arriving at their designated target partitions in T.

Because M' is an exact 3-dimensional matching, the required edges between the vertex gadgets and $\Omega_{M'}$ are guaranteed to exist at each timestep. Therefore, all movements occur without collision, yielding a feasible schedule of makespan $\mu \leq 4$. □

Claim 2. *If the reduced instance $\mathcal{I}'$ is a YES instance, then the original instance $\mathcal{I}$ of* 3-Dimensional Matching *is also a YES instance.*

Proof of Claim 2. To prove this claim, we will first show that μ is exactly four and any feasible schedule with $\mu = 4$ should strictly follow the unique start-target assignment shown in Table 4.1 for all worker robots. We will then show that this unique routing forces the obstacle robots to move in a way that gives a valid 3-D matching in $\mathcal{I}$.

First, we prove that $\mu = 4$. Since $\mathcal{I}'$ is a YES instance, there exists a feasible schedule of makespan $\mu \leq 4$. Assume there exists a schedule with $\mu \leq 3$. Consider worker robots initially placed at S_4. The closest target partition is T_1, which is four steps away. Therefore, we need at least 4 time steps to move them to their destination, which contradicts the assumption. Thus $\mu = 4$.

Now we prove that routing must exactly match the unique assignments previously detailed in Table 4.1. We prove this using a strict process of iterative elimination based on the structure of the input instance.

Since the makespan is bounded by four, all worker robots should have minimum arrival time $\tau \leq 4$. At each step of iterative elimination, certain clusters have exactly one remaining valid target. Then, we assign this unique target cluster to that start cluster.

- **Step 1:** We first identify source partitions that have exactly one target partition reachable with an arrival time $\tau \leq 4$. Based on the graph's structure, robots in S_4, S_7, S_9, S_{10} each have exactly one cluster reachable in $\tau \leq 4$. This strictly forces the following assignments: $S_4 \to T_1$, $S_7 \to T_4$, $S_9 \to T_{12}$, and $S_{10} \to T_7$.

- **Step 2:** Once these initial targets are assigned, they are no longer viable options for other robots. The viable options are reduced to $T \setminus \{T_1, T_4, T_7, T_{12}\}$. There is a new set of robots with only one possible target. For instance, robots at S_{15} could reach T_{12} with $\tau = 3$, but because T_{12} has to be assigned to S_9, S_{15} must move to T_{13}. The following assignments can be fixed: $S_3 \to T_2$, $S_6 \to T_5$, and $S_{15} \to T_{13}$.

- **Step 3:** Now the available target vertices are $T \setminus \{T_1, T_2, T_4, T_5, T_7, T_{12}, T_{13}\}$. The following assignments are fixed: $S_2 \to T_3$ and $S_5 \to T_9$.

- **Step 4:** Continuing this elimination, the remaining target clusters strictly forces $S_8 \to T_6$, $S_{12} \to T_{10}$, and $S_{14} \to T_{14}$. Finally, the last remaining source clusters are assigned to their target clusters: $S_{11} \to T_{11}$, $S_1 \to T_8$, and $S_{13} \to T_{15}$.

Since the assignment of targets in steps 2, 3, and 4 is a consequence assigning only available options in the previous steps, any deviation from these forced assignments would leave at least one cluster without a reachable target, forcing the makespan to exceed 4. This cascade of assignments proves that the start-target assignment for the worker robots shown in Table 4.1 is the only possible assignment for a feasible schedule.

Clearly, the minimum arrival time $\tau = 4$ for all the worker robots. Thus no worker robot can wait at any timestep and must move continuously toward their targets.

As established by the required paths in Table 4.1, worker robots from S_1, S_5, and S_9 must all traverse the obstacle gadget Ω to reach their targets. Because no robot can wait, these three groups will arrive at and occupy the obstacle gadget at consecutive time steps.

To make room for the first set of worker robots from S_1 at $t = 1$, we must move q obstacle robots out of Ω. The only free vertices adjacent to Ω available for moving these obstacle robots is V_5. Let us call the subset of obstacle vertices vacated by these robots $\Omega_{M'}$.

Once these q obstacle robots move to V_5, they cannot return to Ω, nor can any other obstacle robots be moved out, because all neighboring vertices $N(\Omega)$ are occupied by worker robots moving towards their target. Therefore, the worker robots from S_1, S_5, and S_9 must successively move through the exact same set of q empty vertices, $\Omega_{M'}$. The vacated obstacle robots must continuously move toward V_6 to prevent collision with the trailing S_1 robots.

Let $M' \subseteq M$ be the set of q tuples corresponding to the vertices in $\Omega_{M'}$.

- At $t = 1$, S_1 moves into $\Omega_{M'}$. For this to be physically possible without collisions, every vertex in S_1 must have an edge connecting to $\Omega_{M'}$. Since S_1 corresponds to the elements of set A, every element $a_i \in A$ appears in at least one tuple of M'.

- At $t = 2$, S_1 vacates $\Omega_{M'}$, and robots from V_1 (originating from S_5) enter it. Thus, V_1 must have perfect matching to $\Omega_{M'}$, meaning all elements of B are covered by M'.

- At $t = 3$, V_1 vacates $\Omega_{M'}$, and robots from V_2 (originating from S_9) enter it. Thus, V_2 must have perfect matching to $\Omega_{M'}$, meaning all elements of C are covered by M'.

To prove that M' constitutes an exact matching, we examine the cardinality of these sets. The obstacle space $\Omega_{M'}$ holds exactly q vertices, which signifies that the subset M' consists of exactly q triples. The sets A, B, and C contain exactly q distinct elements. Since exactly $3q$ elements are entirely covered within the q triples, redundancy of an element is not possible. Therefore, each element in $A \cup B \cup C$ must appear in exactly one triple, proving that M' is an exact 3-dimensional matching. □

By Claim 1, a valid 3-dimensional matching in $\mathcal{I}$ guarantees a collision-free schedule for $\mathcal{I}'$ with a makespan of exactly $\mu = 4$. Conversely, Claim 2 establishes that any feasible schedule achieving this makespan strictly enforces a unique, continuous routing that maps back to an exact matching in $\mathcal{I}$. Since this reduction runs in polynomial time, we conclude that UGCMP-μ is strongly NP-hard, even for instances with a makespan of $\mu = 4$. □

Corollary 4.1.1. UGCMP-μ is NP-hard when the goal is to minimize average arrival time τ_{av}, even for $\tau_{av} = 4$.

Proof. This follows directly from the reduction in Theorem 4.1. In our construction, a valid solution of $\mu = 4$ requires all k worker robots to reach their destinations in exactly four time steps. Thus $\tau = 4$ for all robots. Consequently, the reduced instance is a YES instance minimizing the average arrival time $\tau_{av} \leq 4$. Therefore, the exact same reduction holds for the average arrival time objective. □

4.3 Polynomial Time Algorithm for $\mu = 1$

In this section, we present an algorithm that solves UGCMP-μ instances with $\mu = 1$ in polynomial time, hence proving this problem is tractable. We provide the proof of correctness and time complexity analysis of the algorithm.

Theorem 4.2. UGCMP-μ *admits a polynomial-time algorithm when* $\mu = 1$.

Proof. We prove this by reducing an instance of UGCMP-μ with $\mu = 1$ to a single-commodity integer maximum flow problem on a flow network H in polynomial time, as shown in Algorithm 2. A generic illustration of the resultant flow network is shown in Figure 4.2. Given an instance $(G, S, T, \Omega, \mu = 1)$ of UGCMP-μ where $|S| = k$ and $|\Omega| = m$, we construct flow network H as follows.

We first define an edge set $E' \subseteq V \times V$ called the **self-loop set** beyond the original graph edges E. It contains a loop (v, v) for each worker robot initially on $s \in (S \cap T)$ and each obstacle initially on $o \in (\Omega \setminus T)$.

The vertex set of H consists of five parts:

- A super source α and a super sink β.
- $P_1 = S \cup \Omega$
- $P_2 = \{e_{\text{in}} \mid e \in E \cup E'\}$
- $P_2' = \{e_{\text{out}} \mid e \in E \cup E'\}$
- $P_3 = T \cup \Omega \cup N(\Omega)$ where $N(\Omega)$ is the set of all neighboring vertices of each $o \in \Omega$.

We add the following directed edges, each with a capacity of 1:

- $\alpha \to P_1$**:** Add an edge (α, v) for each $v \in P_1$.
- $P_3 \to \beta$**:** Add an edge (v, β) for each $v \in P_3$.
- $P_2 \to P_2'$**:** The vertex corresponding to $e_{\text{in}} \in P_2$ has an outgoing edge to the vertex corresponding to the same edge $e_{\text{out}} \in P_2'$.
- **Worker routing:** For each $s \in S$ and $t \in N(s) \cap T$ (as well as $t = s$ if $(s, s) \in E'$), add an edge from $s \in P_1$ to $(s, t)_{\text{in}} \in P_2$, and from $(s, t)_{\text{out}} \in P_2'$ to $t \in P_3$.
- **Obstacle routing:** For each $o \in \Omega$ and $w \in (o \cup N(o)) \setminus T$, add an edge from $o \in P_1$ to $(o, w)_{\text{in}} \in P_2$, and from $(o, w)_{\text{out}} \in P_2'$ to $w \in P_3$.

Algorithm 2: SOLVE-UGCMP-MU1(G, S, Ω, T)

Input: Graph $G = (V, E)$, start vertices S, obstacle positions Ω, target vertices T

Output: **true** iff all worker robots can reach from S to T in ≤ 1 step

```
// 1. Pre-processing and Partitions
```
$k \leftarrow |S|$; $\quad m \leftarrow |\Omega|$;

$E' \leftarrow \{(v, v) \mid v \in (\Omega \setminus T) \cup (S \cap T)\}$; `// Self-loops for staying put`

$P_1 \leftarrow S \cup \Omega$; $\quad P_2 \leftarrow \{\, e_{\text{in}} \mid e \in E \cup E' \}$; $\quad P_2' \leftarrow \{\, e_{\text{out}} \mid e \in E \cup E' \}$;

$P_3 \leftarrow T \cup \Omega \cup N(\Omega)$;

```
// 2. Network Construction
```
Initialize graph H with super-source α and super-sink β;

foreach $v \in P_1$ **do** `// Edges from super source to sources`

  AddEdge$(\alpha,\ v,\ \text{cap} = 1)$;

foreach $e \in E \cup E'$ **do** `// Connecting edge in to edge out`

  AddEdge$(e_{\text{in}},\ e_{\text{out}},\ \text{cap} = 1)$;

foreach $s \in S$ **do** `// Worker routing`

  foreach $t \in N(s) \cap T$, *and* $t = s$ *if* $(s, s) \in E'$ **do**

    AddEdge$(s,\ (s,t)_{\text{in}},\ \text{cap} = 1)$;

    AddEdge$((s,t)_{\text{out}},\ t,\ \text{cap} = 1)$;

foreach $o \in \Omega$ **do** `// Obstacle routing`

  foreach $w \in N(o) \setminus T$, *and* $w = o$ *if* $(o, o) \in E'$ **do**

    AddEdge$(o,\ (o,w)_{\text{in}},\ \text{cap} = 1)$;

    AddEdge$((o,w)_{\text{out}},\ w,\ \text{cap} = 1)$;

foreach $v \in P_3$ **do** `// Edges from sinks to super sink`

  AddEdge$(v,\ \beta,\ \text{cap} = 1)$;

```
// 3. Computation
```
$f^* \leftarrow \text{MAXFLOW}(H, \alpha, \beta)$;

return $f^* = k + m$

The number of vertices in P_1 and P_3 is in $\mathcal{O}(|V|)$, and P_2 and P_2' are in $\mathcal{O}(|V| + |E|)$ since $|E'| \leq |V|$. The number of edges in H is in $\mathcal{O}(|V| + |E|)$ as well, since there can be a maximum of $2(|E| + |E'|)$ edges between P_1&P_2

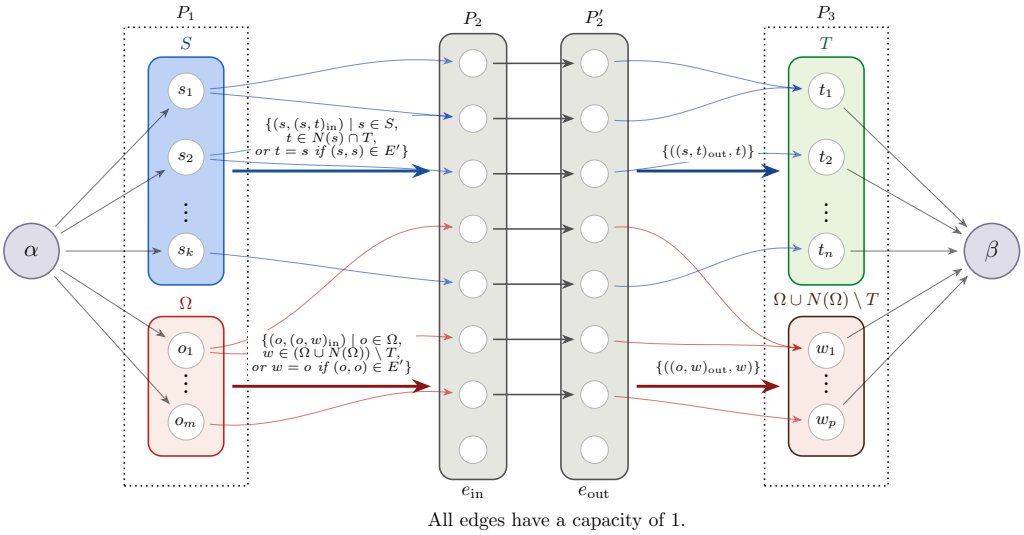


Figure 4.2: Generic illustration of the flow network H constructed for UGCMP-μ when $\mu = 1$. Edge gadgets ($P_2 \to P_2'$) guarantee edge-collision avoidance, while the unit capacities from P_3 the super-sink β prevent vertex collisions.

and P_2'&P_3. Thus, H is of polynomial size with respect to the input graph G. Using the Edmonds-Karp Algorithm, we can find the max-flow on H in $\mathcal{O}(|V_H| \cdot |E_H|^2)$, which is equal to $\mathcal{O}((|V| + |E|)^3)$.

Claim. *The given instance of* UGCMP-μ *($\mu = 1$) is a YES instance if and only if the max flow on H, $f_H^* = k + m$.*

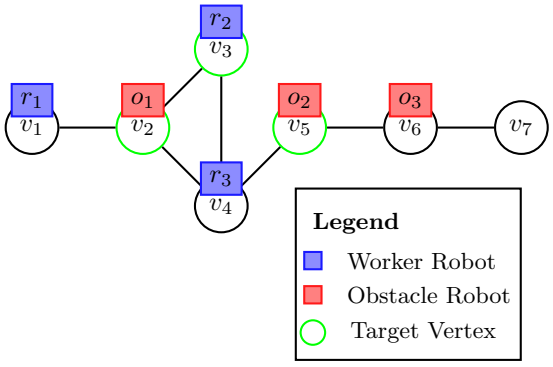
Proof of Claim. $\implies$: Since the UGCMP-μ instance is a YES instance, there exists a valid, collision-free 1-step schedule where all k worker robots reach distinct targets in T, and all m obstacles reach valid non-target vertices in $V \setminus T$. For every robot (worker or obstacle) moving from its start vertex u to its destination v, we push 1 unit of flow along the path $\alpha \to u \to (u,v)_{\text{in}} \to (u,v)_{\text{out}} \to v \to \beta$. Because the schedule is collision-free, no two robots share the same edge (satisfying the capacity of 1 on $(u,v)_{\text{in}} \to (u,v)_{\text{out}}$) and no two robots arrive at the same destination vertex (satisfying the capacity of 1 on $v \to \beta$). Since all k workers and m obstacles successfully route to distinct valid endpoints, exactly $k + m$ units of flow reach the super sink β. Thus,

$f_H^* = k + m$.

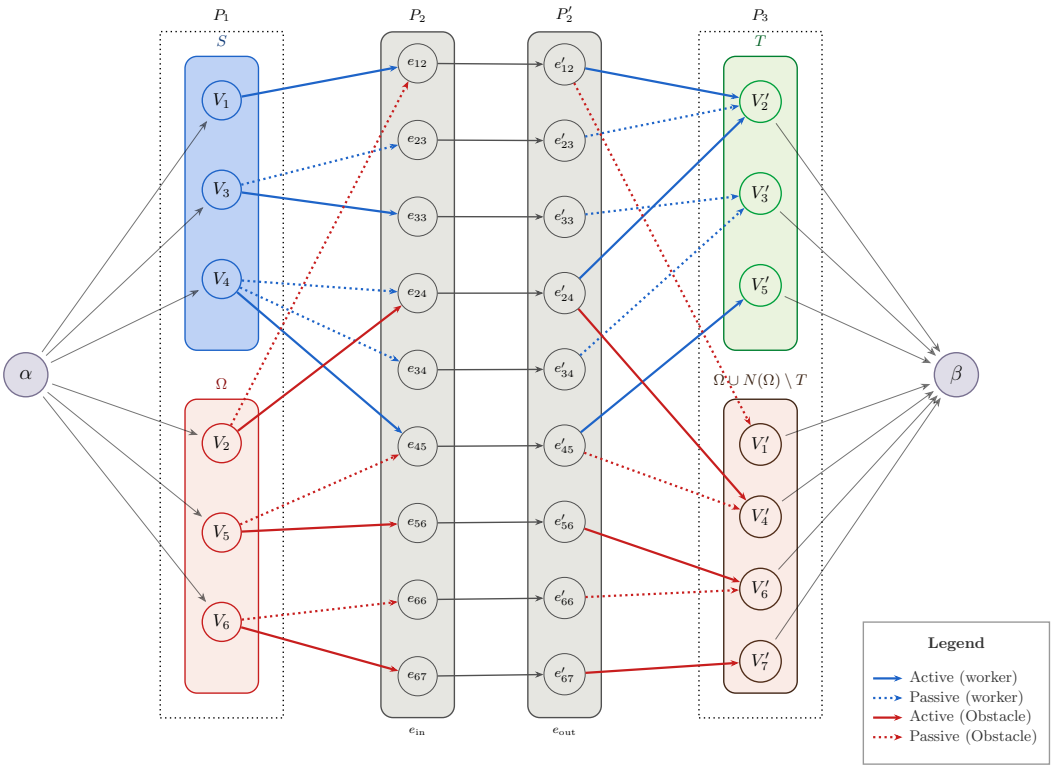
$\Longleftarrow$ **:** If $f_H^* = k + m$, by the Integral Flow Theorem, there exists an integer maximum flow of this value. Since the outgoing edges from the super source α have a capacity of 1, exactly 1 unit of integer flow originates from each starting position in $P_1 = S \cup \Omega$. This flow must traverse exactly one edge gadget $(u, v)_{\text{in}} \to (u, v)_{\text{out}}$ to reach P_3. The capacity of 1 on the edge gadgets ensures that no two robots traverse the same physical edge in G, preventing edge collisions. The flow then moves from P_3 to the super sink β. The capacity of 1 on these edges ensures that exactly one robot arrives at any given destination vertex, preventing vertex collisions. Finally, the network construction guarantees that flow originating from S can only route to target vertices T, while flow originating from Ω can only route to non-target vertices $(V \setminus T)$. Because all k workers arrive at T, all m obstacles move out of their way, and no collisions occur, the flow paths directly map to a valid 1-step schedule, making it a YES instance. □

Therefore, the maximum flow directly computes a valid schedule when $\mu = 1$, establishing that the problem is solvable in polynomial time. □

To illustrate this reduction, Figure 4.3 demonstrates a concrete UGCMP-μ instance alongside its corresponding flow network H. For the input instance in Figure 4.3a, the resultant flow network is shown in Figure 4.3b. The movement of a unit flowing through the vertices corresponding to S and Ω in P_1 will now dictate the routing of the robot starting from that vertex in Instance (G, S, T, Ω, μ). A unit of flow directed from $\alpha \to V_1 \to e_{1,2} \to e'_{1,2} \to V'_2 \to \beta$ corresponds to robot r_1 moving from v_1 to v_2 through (v_1, v_2) in (G, S, T, Ω, μ). The solid edges in Figure 4.3b shows edges in the network through which the flow happens, and dashed edges are unused. In this case, Algorithm 2 will return true because $f* = 6 = k + m$, and the final configuration will be: $r_1 \to v_2, r_2 \to v_3, r_3 \to v_5, o_1 \to v_4, o_2 \to v_6,$ and $o_3 \to v_7$.



(a) Input Instance $(G, S, T, \Omega, \mu = 1)$



(b) Flow Network H

Figure 4.3: Reduction of a UGCMP-μ instance to a maximum flow problem for makespan $\mu = 1$. Figure 4.3a is an instance on a graph G with $k = 3$ workers (S) and $m = 3$ obstacles (Ω), and target destinations $(\mathcal{T})$. Figure 4.3b shows the resultant unit-capacity flow network H. Solid edges indicate the active flow paths and $f^* = k + m = 6$, corresponding to a valid 1-step routing schedule. In the resultant final configuration of UGCMP-μ instance: $r_1 \to v_2, r_2 \to v_3, r_3 \to v_5, o_1 \to v_4, o_2 \to v_6,$ and $o_3 \to v_7$.

Chapter 5

Tree-UGCMP-ℓ with One Obstacle

In this chapter, we present a preliminary analysis of the UGCMP Problem on a tree graph with exactly one obstacle, with the objective of minimizing the total routing length ℓ (i.e., the sum of the lengths of all robot paths). We denote this specific variant as Tree-UGCMP-ℓ with One Obstacle.

Let the underlying graph be a tree $G_T = (V, E)$. Suppose the single obstacle robot initially occupies the vertex $v_r \in V$. We fix the root of G_T at v_r. Removing v_r partitions the tree into a set of disjoint connected components $G_T[\mathcal{K}_1], G_T[\mathcal{K}_2], \ldots, G_T[\mathcal{K}_p]$, each rooted at a child $v_{r,i}$ of v_r.

We say a subtree $G_T[\mathcal{K}_i]$ is *balanced* if it contains an equal number of sources and sinks; otherwise, it is *unbalanced*. The structure of these components relative to the obstacle's starting position plays a significant role in the optimal movement strategy.

5.1 Robot Motion on a Balanced Subtree

Lemma 1. For every balanced subtree $G_T[\mathcal{K}_i]$, there exists a solution that minimizes total length in which no robot moves through the edge $(v_r, v_{r,i})$.

Proof. Assume that in some optimal solution, a robot A with source $s_1 \in G_T[\mathcal{K}_i]$ is assigned to a sink $t_1 \notin G_T[\mathcal{K}_i]$, thus crossing the edge $(v_{r,i}, v_r)$ in

the upward direction.

Since $G_T[\mathcal{K}_i]$ is balanced, the number of sources and sinks within $G_T[\mathcal{K}_i]$ are equal. Robot A moving to $t_1 \notin G_T[\mathcal{K}_i]$ leaves an unmatched sink $t_2 \in G_T[\mathcal{K}_i]$. Since $G_T[\mathcal{K}_i]$ is balanced, the complement $G_T \setminus G_T[\mathcal{K}_i]$ is also balanced. Hence there must exist a source $s_2 \notin G_T[\mathcal{K}_i]$ whose assigned robot B is routed to $t_2 \in G_T[\mathcal{K}_i]$, crossing the edge $(v_r, v_{r,i})$ in the downward direction. The paths of A and B in this solution are:

$$A : s_1 \ \to \ \cdots \ \to \ v_{r,i} \ \to \ v_r \ \to \ \cdots \ \to \ t_1,$$
$$B : s_2 \ \to \ \cdots \ \to \ v_r \ \to \ v_{r,i} \ \to \ \cdots \ \to \ t_2.$$

Now swap the sink assignments of A and B: route A to $t_2 \in G_T[\mathcal{K}_i]$ and B to $t_1 \notin G_T[\mathcal{K}_i]$. The resulting paths are:

$$A' : s_1 \ \to \ \cdots \ \to \ v_{r,i} \ \to \ \cdots \ \to \ t_2,$$
$$B' : s_2 \ \to \ \cdots \ \to \ v_r \ \to \ \cdots \ \to \ t_1.$$

In the rerouted solution, robot A' travels entirely within $G_T[\mathcal{K}_i]$ (or at most reaches $v_{r,i}$), and robot B' travels entirely outside $G_T[\mathcal{K}_i]$. Neither robot crosses the edge $(v_r, v_{r,i})$. All other segments of both paths are retained exactly as before. Therefore, the total length decreases by exactly two since each of A and B previously traversed this edge once and neither does so in the rerouted solution.

This reduction in total length contradicts the assumption that the original assignment was optimal. Hence, no optimal solution can assign a robot from a source inside $G_T[\mathcal{K}_i]$ to a sink outside $G_T[\mathcal{K}_i]$, and consequently, the edge $(v_r, v_{r,i})$ is never traversed in any length-optimal solution. □

5.2 Reduction Rules

To simplify the instance before routing the obstacle, we can apply the following reduction rule to resolve isolated assignments towards the leaf end of the tree.

Reduction Rule 5.1. Suppose there exists a sink t_1 which is a leaf node and a source s_1 in a connected component $G_T[\mathcal{K}_i]$. We can move the robot on s_1 to t_1 along the unique path in G_T, and delete the node t_1 from the instance.

5.3 Case Analysis of Obstacle Movement

Based on the distribution of balanced and unbalanced components, we can categorize the required movements of the obstacle robot to achieve a length-optimal schedule. A summary is visualized in Figure 5.1.

1. **Every connected component is balanced:** The obstacle robot does not need to move (by Lemma 1). Each component $G_T[\mathcal{K}_i]$ is solved independently within itself, since every source-destination pair can be matched without crossing into another component.

2. **There exists at least one balanced connected component $G_T[\mathcal{K}_i]$:** The balanced component is solved internally. The obstacle must move to make way for robots in the unbalanced components.

   (a) **The root $v_{r,i}$ of the balanced subtree is *not* a sink:** The obstacle robot needs to move exactly once (into $v_{r,i}$ to clear v_r).

   (b) **The root $v_{r,i}$ of the balanced subtree *is* a sink but *not* a source:** The obstacle will have to move at least twice.

      i. v_r **is *not* a sink:** The obstacle needs to move exactly twice.

      ii. v_r ***is*** **a sink:** The obstacle needs to move at least twice.

   (c) **The root $v_{r,i}$ of the balanced subtree *is* a sink and a source:** If $v_{r,i}$ is a leaf node, then we cannot move the obstacle robot to $v_{r,i}$. Otherwise, check if the subtrees of the balanced component $G_T[\mathcal{K}_i]$ are balanced or not.

      i. If balanced, then the robot at $v_{r,i}$ doesn't need to move unless to make way for the obstacle.

ii. If unbalanced, the robot at $v_{r,i}$ needs to move at least once. Then, if v_r is not a sink, the obstacle robot needs to move exactly twice. If v_r is a sink, the obstacle robot needs to move at least twice.

3. **Every connected component is unbalanced:** The obstacle robot must move at least twice, as there is no single safe balanced branch it can permanently occupy while all components exchange robots through the root.

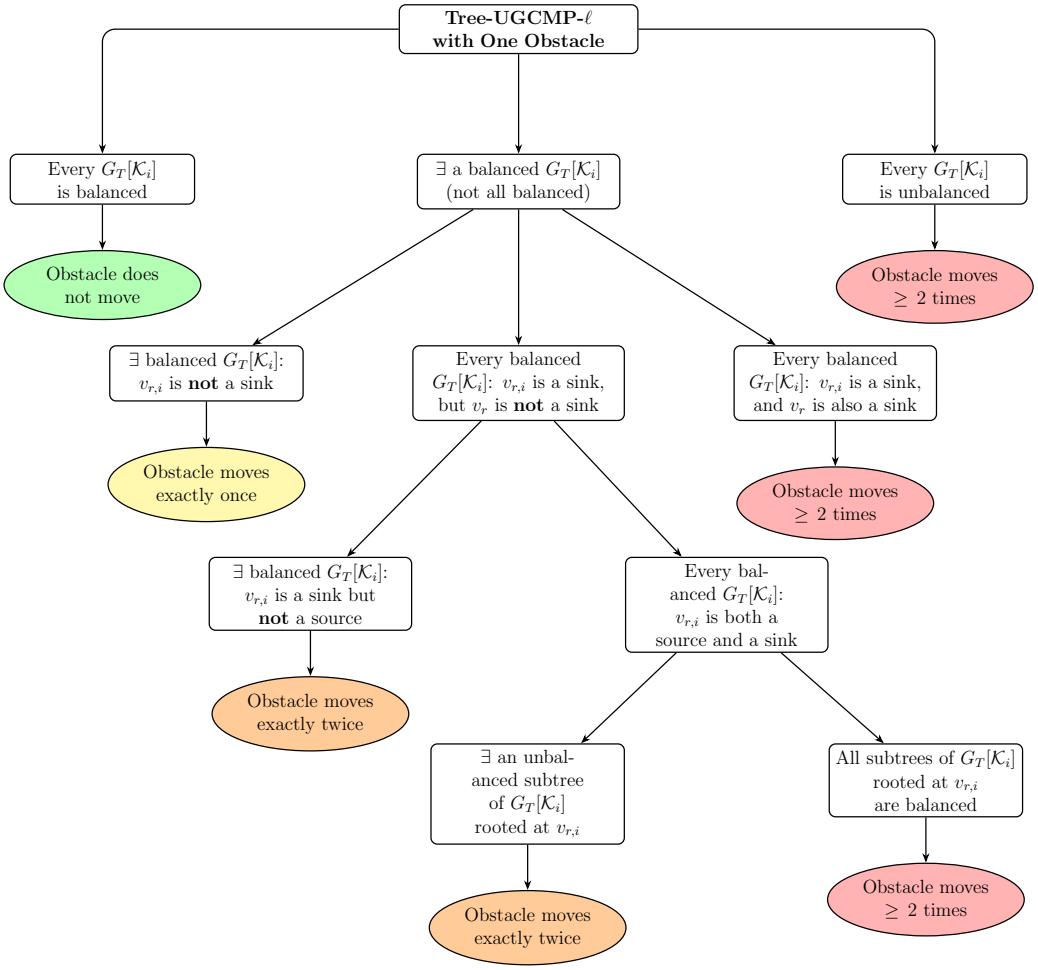


Figure 5.1: Case analysis for Tree-UGCMP-ℓ with One Obstacle. $G_T[\mathcal{K}_i]$ denotes the subtree rooted at child $v_{r,i}$ of the obstacle's starting vertex v_r.

Chapter 6

Conclusions

6.1 Results and Discussion

In this thesis, we explored the computational complexity of GCMP, focusing on the implications of introducing obstacle robots into both labeled and unlabeled variants.

For the labeled variant with the objective of length optimization LGCMP-ℓ, we presented an $n^{O(k+m)}$ algorithm. This result successfully places the problem in the complexity class XP when parameterized by the total number of robots $k+m$, establishing that the problem is solvable in polynomial time when the total number of robots is a constant. However, we note that an XP classification is computationally weaker than Fixed-Parameter Tractability, as the exponent of the polynomial heavily depends on the parameter, limiting its practical scalability for larger multi-agent systems.

For the unlabeled variant with the objective of makespan minimization UGCMP-μ, which is known to be highly tractable in the absence of obstacles, we identified it becomes computationally challenging when optimizing for time (makespan). We proved that UGCMP-μ is NP-hard for a makespan of $\mu = 4$ (and average arrival time $\tau_{av} = 4$). This demonstrates that the interchangeability of worker robots is insufficient to overcome the challenges created by obstacle robots.

In contrast to this this general intractability, we prove that the problem

is in P when $\mu = 1$. In the classical obstacle-free unlabeled CMP the problem is solvable by reducing to a time expanded single-commodity network flow problem precisely because schedules can be constructed with only unidirectional movement through edges, allowing the scope of makespan to be strictly bounded. Introducing obstacle robots typically breaks this tractability in two ways: it transforms the routing into a multi-commodity problem, and it forces opposing edge traversals to clear paths, invalidating standard time bounds. However, when $\mu = 1$, these two structural issues are bypassed. First, because there are no intermediate time steps, the sets of possible target vertices for workers and obstacles do not collide, allowing us to maintain disjoint sets and avoid multi-commodity path conflicts. Second, the single-step condition trivially bounds the time and strictly prevents any edge from being traversed in opposite directions. Consequently, the problem circumvents the complexities usually introduced by obstacles and remains efficiently solvable.

Finally, we analyzed UGCMP for distance optimization on tree graphs containing a single obstacle. By examining the balance of sources and sinks within discrete subtrees, we did a case analysis, demonstrating topological restrictions might simplify the effect of obstacles.

6.2 Future Work

While this thesis resolves some open questions regarding the complexity of GCMP, it also highlights areas for future research. We propose the following directions:

- **Finer Parameterized Analysis of LGCMP-ℓ:**

  While we established an XP algorithm for distance optimization parameterized by $k + m$, it remains an open question whether an FPT algorithm exists for this parameter alone.

- **Closing the Tractability Gap in UGCMP-μ:**

  We established that UGCMP-μ is in P for $\mu = 1$ and NP-hard for $\mu = 4$. The exact computational complexity for makespans $\mu = 2$

and $\mu = 3$ remains an open question. Determining the exact point at which the problem transitions from tractable to intractable will provide a deeper understanding of the problem's core difficulty.

- **Constant Number of Obstacles:**

  Our NP-hardness results for UGCMP-μ utilize arbitrary numbers of obstacle robots. It is worth investigating whether the time optimization problem becomes tractable if the number of obstacle robots m is restricted to a small, fixed constant while the number of worker robots k remains arbitrary.

- **Restricted Graph Classes:**

  Our structural analysis on trees was partial and limited to length optimization with a single obstacle. Completing and extending our tree analysis to handle multiple obstacles ($m \geq 2$) is also a further direction to move in. Future research should analyze UGCMP-μ and UGCMP-ℓ on restricted graph classes such as trees or grids.

Bibliography

[1] 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"," *The International Journal of Robotics Research*, vol. 3, no. 4, pp. 76–88, 1984.

[2] O. Goldreich, "Finding the shortest move-sequence in the graph-generalized 15-puzzle is NP-hard," in *Studies in Complexity and Cryptography: Miscellanea on the Interplay between Randomness and Computation*, pp. 1–5, Berlin, Heidelberg: Springer-Verlag, 2011.

[3] P. Surynek, "An optimization variant of multi-robot path planning is intractable," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, (Atlanta, Georgia, USA), pp. 1261–1263, AAAI Press, 2010.

[4] J. Yu and S. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, pp. 1443–1449, Jun. 2013.

[5] D. Ratner and M. Warmuth, "The (n^2-1)-puzzle and related relocation problems," *J. Symb. Comput.*, vol. 10, pp. 111–137, July 1990.

[6] D. Kornhauser, G. Miller, and P. Spirakis, "Coordinating pebble motion on graphs, the diameter of permutation groups, and applications," in *25th Annual Symposium onFoundations of Computer Science, 1984.*, pp. 241–250, 1984.

[7] G. Goraly and R. Hassin, "Multi-color pebble motion on graphs," *Algorithmica*, vol. 58, pp. 610–636, Nov 2010.

[8] J. Yu, "Diameters of permutation groups on graphs, multi-agent path planning, and integer multiflow on directed acyclic graphs," *CoRR*, vol. abs/1205.5263, 2012.

[9] C. H. Papadimitriou, P. Raghavan, M. Sudan, and H. Tamaki, "Motion planning on a graph," in *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 511–520, IEEE, 1994.

[10] A. Deligkas, E. Eiben, R. Ganian, I. Kanj, and M. S. Ramanujan, "Parameterized algorithms for coordinated motion planning: Minimizing energy," in *Proceedings of the 51st International Colloquium on Automata, Languages, and Programming (ICALP)*, vol. 297 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 53:1–53:19, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024.

[11] A. Felner and R. Stern, "Multi-agent path finding with unassigned agents (MAPFUA)," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 40, pp. 39691–39698, 2026.

[12] J. Yu and S. M. LaValle, "Multi-agent path planning and network flow," *CoRR*, vol. abs/1204.5717, 2012.

[13] E. Eiben, R. Ganian, and I. Kanj, "The parameterized complexity of coordinated motion planning," in *39th International Symposium on Computational Geometry (SoCG 2023)*, vol. 258 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pp. 28:1–28:16, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023.

[14] F. Fioravantes, D. Knop, J. M. Křišťan, N. Melissinos, and M. Opler, "Exact algorithms and lowerbounds for multiagent pathfinding: Power of treelike topology," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, pp. 17129–17137, 2024.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 3rd ed., 2009.

[16] J. Kleinberg and É. Tardos, *Algorithm Design*. Addison-Wesley, 2005.

[17] M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh, *Parameterized Algorithms*. Texts in Computer Science, Springer, 2015.