TU WIEN Informatics

The Complexity of Routing Few Robots in a Crowded Network

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Logic and Computation

eingereicht von

B.Sc. Dominik Leko
Matrikelnummer 51823222

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ass. Professor Robert Ganian, PhD

Wien, 5. September 2025

Dominik Leko	[redacted] Robert Ganian

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Informatics

The Complexity of Routing Few Robots in a Crowded Network

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Logic and Computation

by

B.Sc. Dominik Leko
Registration Number 51823222

to the Faculty of Informatics

at the TU Wien

Advisor: Ass. Professor Robert Ganian, PhD

Vienna, September 5, 2025

Dominik Leko

[redacted]
Robert Ganian

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Erklärung zur Verfassung der Arbeit

B.Sc. Dominik Leko

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, haben ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT- Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 5. September 2025

Dominik Leko

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Danksagung

Mein besonderer Dank gilt meinem Betreuer Robert Ganian, der sich immer Zeit genommen hat, um meine Abschlussarbeit mit mir zu besprechen, mich in seine Forschung einbezogen hat, meinen Ideen Gehör geschenkt hat und während des gesamten Prozesses stets entspannt und unterstützend war. Als ich sah, wie einige Kommilitonen sogar Schwierigkeiten hatten, ihre Betreuer zu erreichen, war ich besonders dankbar, einen so engagierten und zugänglichen Betreuer zu haben.

Ich möchte auch meinen Co-Autoren der Arbeit danken, die mich zu dieser Abschlussarbeit motiviert hat: Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj und M. S. Ramanujan. Unsere Zusammenarbeit war kurz, aber intensiv und von gegenseitigem Respekt geprägt.

Schließlich möchte ich meiner Familie und meinen Freunden danken, die mir geholfen haben, motiviert zu bleiben.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Acknowledgements

I would like to give special thanks to my supervisor, Robert Ganian, for always making time to discuss my thesis, for including me in his research, for hearing out my ideas, and for being overall relaxed and supportive throughout the process. Watching some fellow students struggle even to reach their supervisors made me especially grateful to have such an engaged and approachable one.

I also want to thank my co-authors on the paper that motivated this thesis: Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan. Our collaboration was brief but engaging and respectful.

Finally, I want to thank my family and my friends, who helped me stay motivated.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Kurzfassung

Das Steuern von Robotern in einer Umgebung unter Vermeidung von Kollisionen ist ein häufiges Problem in der Industrie. Einige natürliche Formalisierungen dieses Problems modellieren Roboter als zweidimensionale Quadrate oder Scheiben, die sich durch einen „Raum" in der Ebene bewegen. Diese Arbeit betrachtet ein abstrakteres, graphentheoretisches Modell, bei dem die Roboter die Knoten eines Graphen einnehmen und sich entlang seiner Kanten bewegen. Wir entwickeln den Stand der Technik von zwei gängigen Problemformalisierung namens GCMP und GCMP1 weiter. Wir untersuchen das Problem unter dem Gesichtspunkt der parametrisierten Komplexität und führen zwei neue Parametrisierungen ein: eine, die einen W[1]-Härtebeweis liefert, und eine andere, die die Einbettung in FPT festlegt.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Routing robots in an environment while avoiding collisions is a common problem in industry. Some natural formalizations of this problem model robots as two-dimensional squares or discs that navigate through a "room" in the plane. This thesis considers a more abstract, graph-theoretic model, where the robots occupy the vertices of a graph and move around along its edges. We advance the state of the art of a common problem formalization, called GCMP and GCMP1. We study the problem through the lens of parameterized complexity and introduce two new parameterizations: one yielding a W[1]-hardness proof, and the other establishing inclusion in FPT.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Contents

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 1

Introduction

The task of routing a set of robots (agents) from a set of starting positions to their destinations while avoiding collisions appears in several real-world settings. Just consider an automated warehouse in which goods must be constantly relocated, or a group of search-and-rescue robots searching for life in a broken building or drones creating a light show in the sky. For these instances efficient scheduling algorithms are highly beneficial, ideally with performance guarantees. The most prominent optimization goals for these algorithms are to minimize (i) the *makespan*, i.e. the length of the schedule, and (ii) the total *energy* expended (equivalently, the total distance moved) by the robots. Any schedule must be safe, i.e. prevent any collisions between different robots or between robots and the environment. There already exists extensive research work in this direction, particularly from the Computational Geometry [1, 23], Artificial Intelligence [5, 30] and Robotics [3, 22] research communities. Although algorithms optimizing makespan exist [17, 10, 22], this thesis will only deal with energy-minimizing variants.

There are different ways to model the settings described above. One might for example consider the robots to be unit discs living on a 2D-plane, trying to find their way inside of a "room" [4]. However, the formalization we are working with is contained in the field of Graph Theory, and has been studied in several recent papers [9, 10, 17, 18, 20]. We consider the Graph theoretic formalization Graph Coordinated Motion Planning, which we formalize below.

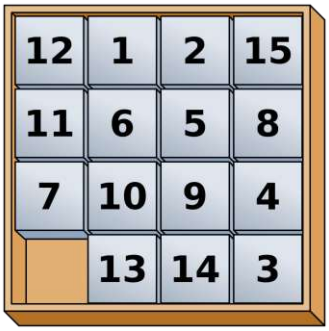

Figure 1.1: The popular 15-puzzle game [31].

Graph Coordinated Motion Planning (GCMP)

Input: A tuple $(G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \ell)$, where G is a graph, $\mathcal{R} = \{R_i \mid i \in \mathbb{N}\}$ is a set of robots partitioned into sets $\mathcal{M}$ and $\mathcal{F}$, where each robot in $\mathcal{M}$ is given as a pair of vertices (s_i, t_i) and each robot in $\mathcal{F}$ as a single vertex s_i, and a budget $\ell \in \mathbb{N}$.

Problem: Is there a schedule for $\mathcal{R}$ of total traveled length at most ℓ?

Intuitively, are given a graph G, robots $\mathcal{R}$, partitioned into a set of *marked* robots $\mathcal{M}$ and *free* robots $\mathcal{F}$, and an energy budget ℓ. We also refer to the free robots as *blockers*. Each robot occupies one vertex, and the robots move along the edges to neighboring vertices. Only one robot may sit on a vertex at a time and two robots may not use the same edge at the same time. The goal is to route the marked robots to their destinations, while avoiding collisions with other (marked or free) robots. The robots move in concrete time steps, and it requires one time step to reach a vertex u from a neighboring vertex v. A schedule is a sequence of commands specifying each robot's move at every time step. Graph Coordinated Motion Planning 1 is the restriction of GCMP to instances where $|\mathcal{M}| = 1$.

The precursor formalization to GCMP is called Coordinated Motion Planning (short: CMP), in which all robots have a destination. CMP has been researched extensively der various discrete and continuous settings [4, 5, 11, 16]. For example, Boyarski et al. devised a routing algorithm that finds a route for every marked robot separately and then attempts to consolidate conflicts by "merging" two robots. More recently, people have considered *sliding* moves, where only one robot may move at a time and it costs only one move to move a robot arbitrarily far, provided the path it travels is free of other robots [18]. CMP generalizes the famous $(n^2 - 1) - puzzle$, which can be seen in Figure 1.1. Both CMP and GCMP1 have been shown to be NP-hard [12, 28, 32].

Our problem, GCMP, generalizes CMP by allowing arbitrary partitions of robots into a marked and a free set. The classical complexity of several of its variants has been settled already in the 90's [27]. So, instead, we inspect the problem from the lens of

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

parameterized complexity. Several results for CMP in this regard are already known [17], and so our aim is to continue research on GCMP and GCMP1. In particular we follow up on work in the ICALP paper [9], which (amongst other results) shows that:

1. GCMP1 is fixed-parameter tractable parameterized by $|\mathcal{R}|$,
2. GCMP is fixed-parameter tractable parameterized by $|\mathcal{R}|$ plus the treewidth of G, and
3. GCMP is W[1]-hard parameterized by ℓ, but it becomes fixed-parameter tractable on graphs of bounded local treewidth.

One natural direction to continue research is to lift the parameterization on $|\mathcal{R}|$ and instead parameterize by $|\mathcal{M}|$. The hope is that even if the graph is filled with many blockers, the fact that there are only few marked robots allows for an efficient algorithm. This thesis in particular will be dealing with those cases of GCMP where the number of marked robots is one or very small, hence the title. Our first result, formalized in Theorem 1, is a W[1]-hardness proof for GCMP1 parameterized by ℓ. This result directly strengthens the third result from the precursor paper, effectively showing that it is not the number of marked robots that makes the problem difficult.

Theorem 1. GCMP1 *is W[1]-hard when parameterized by* ℓ.

Another direction would be to consider the underlying structure itself: Because we are dealing with a graph, we can parameterize by popular graph measures. In our second result we parameterize by *treedepth*, a graph parameter that is closely related to the theory of sparsity [25]. Adding treedepth to the parameter makes the problem tractable. We formalize the second result in the following Theorem.

Theorem 2. GCMP *is fixed-parameter tractable when parameterized by the number* $k = |\mathcal{M}|$ *of marked robots plus the treedepth of the input graph.*

Finally, we note that the results of this thesis were presented at the Algorithms and Data Structures Symposium (WADS 2025) and appeared in the proceedings of that conference [8]. The exposition of this thesis forms an extended version of those results, with longer explanations for clarity. The research was carried out during a long sequence of meetings with the co-authors of the aforementioned publication, and the author of this thesis actively contributed to all parts of the publication. In particular, the main contribution of the author was towards [8, Theorem 1], which is reflected in Chapter 3 and is accompanied with the most detailed explanation. The secondary contribution was towards [8, Theorem 2], which is covered in Chapter 4.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 2

Preliminaries

All graphs considered in this thesis are undirected and simple. We assume familiarity with standard graph-theoretic concepts and terminology [13] as well as with basic notions in parameterized complexity, including *fixed-parameter tractability* and W[1]-*hardness* [7, 14, 21]. For $n \in \mathbb{N}$, we let $[n]$ and $[n]_0$ denote the sets $\{1, \ldots, n\}$ and $\{0, \ldots, n\}$, respectively.

Parameterized Complexity. A *p*arameterized problem Q is a subset of $\Omega^* \times \mathbb{N}$, where Ω is a fixed alphabet. Each instance of Q is a pair (I, κ), where $\kappa \in \mathbb{N}$ is called the *p*arameter. A parameterized problem Q is *f*ixed-parameter tractable (FPT) [7, 14, 21], if there is an algorithm, called an *FPT-algorithm*, that decides whether an input (I, κ) is a member of Q in time $f(\kappa) \cdot |I|^{\mathcal{O}(1)}$, referred to as *FPT-time*, where f is a computable function and $|I|$ is the input instance size. The class FPT denotes the class of all fixed-parameter tractable parameterized problems.

Showing that a parameterized problem is hard for the complexity classes W[1] or W[2] rules out the existence of a fixed-parameter algorithm under well-established complexity-theoretic assumptions. Such hardness results are typically established via a *parameterized reduction*, which is an analogue of a classical polynomial-time reduction with two notable distinctions: a parameterized reduction can run in time $f(k) \cdot n^{\mathcal{O}(1)}$, but the parameter of the produced instance must be upper-bounded by a function of the parameter in the original instance. We refer to [7, 14] for more information on parameterized complexity.

Treewidth and Treedepth. *Treewidth* is a fundamental graph parameter which can be seen as a measure of how similar a graph is to a tree; trees have treewidth 1, while the complete n-vertex graph has treewidth $n-1$. A formal definition of treewidth will not be necessary to obtain our results; however, we will make use of Courcelle's Theorem [6] (introduced later) which essentially says that problems expressible in a certain fragment

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

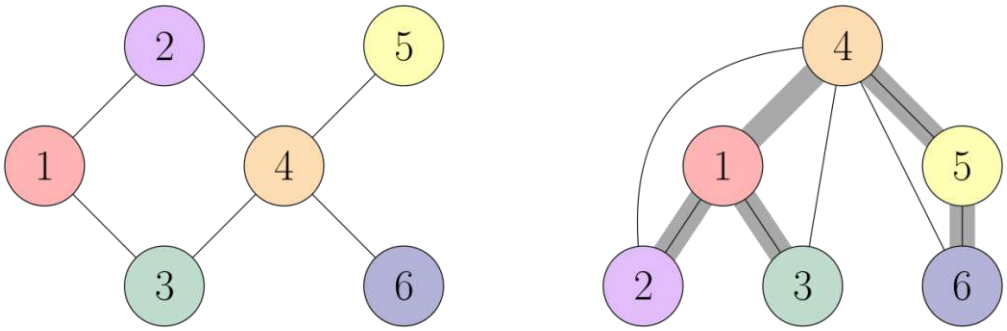

Figure 2.1: A graph G (left) and an optimal forest embedding (right) F. The slim black edges on the right do not technically exist.

of logic can be solved efficiently on graphs of bounded treewidth. Hence, we proceed by defining our other parameter of interest.

Definition 1 (Forest embedding and treedepth)**.** *A* forest embedding *of a graph G is a pair (F, f), where F is a rooted forest and $f : V(G) \to V(F)$ is a bijective function, such that for each $\{u, v\} \in E(G)$, either $f(u)$ is a descendant of $f(v)$, or $f(v)$ is a descendant of $f(u)$. The* depth *of the forest embedding is the number of vertices in the longest root-to-leaf path in F. The* treedepth *of a graph G, denoted by $\mathsf{td}(G)$, is the minimum over the depths of all possible forest embeddings of G. When G is connected, F is a tree and we call it a* tree embedding*.*

To assist the reader's intuition, we devise the example shown in Figure 2.1. Graph G on the left has treedepth 3 and, on the right is an optimal forest embedding. The thick gray lines are the edges in F, while the slim black lines represent edges from G.

Below, we state two facts about treedepth which will be useful for our considerations.

Proposition 1 ([26])**.** *For a graph G and any vertex $v \in V(G)$, it holds that $\mathsf{td}(G) \leq 1 + \max_{i \in [p]} \mathsf{td}(G_i)$, where $G_1, \ldots, G_p$ are the connected components of $G - v$.*

Proposition 2 ([26])**.** *For a graph G, the maximum distance between any two vertices in G is at most $2^{\mathsf{td}(G)}$.*

Monadic Second Order Logic (MSO) of Graphs. A sentence in MSO is built using the logical connectives $\vee$, $\wedge$, $\neg$, $\Leftrightarrow$, $\Rightarrow$, variables for vertices, edges, sets of vertices and sets of edges, and the quantifiers $\forall$ and $\exists$, which can be applied to these variables. The atoms include the following binary relations: (i) $u \in U$, where u is a vertex variable and U is a vertex set variable; (ii) $d \in D$, where d is an edge variable and D is an edge set variable; (iii) $\mathbf{inc}(d, u)$, where d is an edge variable, u is a vertex variable, with the interpretation that the edge d is incident to u; (iv) equality of variables representing vertices, edges, vertex sets and edge sets. We refer to the classical works in the area [2, 6] for a more detailed introduction to MSO.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The well-known Courcelle's Theorem [6, 2] states that checking whether a given graph G models a given MSO formula ϕ can be done in FPT time parameterized by the treewidth of G and the size of ϕ. Moreover, this result holds even if some vertices of G are given labels or colors (i.e., we allow a fixed number of additional unary relations over $V(G)$). This is because one can produce an equivalent graph G' such that G has bounded treewidth if and only if G' does, plus an alternate MSO formula ϕ' such that G models ϕ if and only if G' models ϕ'.

Finally we consider our problem of interest for this thesis, which is called *Graph Coordinated Motion Planning*.

Graph Coordinated Motion Planning (GCMP)

Input: A tuple $(G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \ell)$, where G is a graph, $\mathcal{R} = \{R_i \mid i \in \mathbb{N}\}$ is a set of robots partitioned into sets $\mathcal{M}$ and $\mathcal{F}$, where each robot in $\mathcal{M}$ is given as a pair of vertices (s_i, t_i) and each robot in $\mathcal{F}$ as a single vertex s_i, and a budget $\ell \in \mathbb{N}$.

Problem: Is there a schedule for $\mathcal{R}$ of total traveled length at most ℓ?

In our problems of interest, we are given an undirected graph G and a set $\mathcal{R} = \{R_1, R_2, \ldots, R_k\}$ of k robots where $\mathcal{R}$ is partitioned into two sets $\mathcal{M}$ and $\mathcal{F}$. Each $R_i \in \mathcal{M}$, has a starting vertex s_i and a destination vertex t_i in $V(G)$ and each $R_i \in \mathcal{F}$ is associated only with a starting vertex $s_i \in V(G)$. We refer to the elements in the set $\{s_i \mid i \in [k]\} \cup \{t_i \mid R_i \in \mathcal{M}\}$ as *terminals*. The set $\mathcal{M}$ contains robots that have specific destinations they must reach, whereas $\mathcal{F}$ is the set of remaining "free" robots. We assume that all the s_i's are pairwise distinct and that all the t_i's are pairwise distinct. A vertex $v \in V(G)$ is *free* at time step $x \in [0, t]$ if no robot is located at v at time step x; otherwise, v is *occupied*. We use a discrete time frame $[0, t]$, $t \in \mathbb{N}$, to reference the sequence of moves of the robots and in each time step $x \in [0, t]$, exactly one robot moves from its current vertex to an adjacent free vertex.

A *route* for robot R_i is a tuple $W_i = (u_0, \ldots, u_t)$ of vertices in G such that (i) $u_0 = s_i$ and $u_t = t_i$ if $R_i \in \mathcal{M}$ and (ii) $\forall j \in [t]$, either $u_{j-1} = u_j$ or $u_{j-1}u_j \in E(G)$. Put simply, W_i corresponds to a "walk" in G, with the exception that consecutive vertices in W_i may be identical (representing waiting time steps), in which R_i begins at its starting vertex at time step 0, and if $R_i \in \mathcal{M}$ then R_i reaches its destination vertex at time step t. Two routes $W_i = (u_0, \ldots, u_t)$ and $W_j = (v_0, \ldots, v_t)$, where $i \neq j \in [k]$, are *non-conflicting* if (i) $\forall r \in \{0, \ldots, t\}$, $u_r \neq v_r$, and (ii) $\nexists r \in \{0, \ldots, t-1\}$ such that $v_{r+1} = u_r$ and $u_{r+1} = v_r$. Otherwise, we say that W_i and W_j *conflict*. Intuitively, two routes conflict if the corresponding robots are at the same vertex at the end of a time step or if two robots traverse the same edge in the same time step.

A *schedule* S for $\mathcal{R}$ is a set of pairwise non-conflicting routes $W_i, i \in [k]$, during a time interval $[0, t]$. The (*traveled*) *length* of a route (or its associated robot) within S is the number of time steps j such that $u_j \neq u_{j+1}$. The *total traveled length* of a schedule is

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

the sum of the lengths of its routes; this value is often called the *energy* in the literature (e.g., see [19]).

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 3

A Robot With a Goal

In this chapter, we use a *parameterized reduction* to show that GCMP1 parameterized by ℓ is W[1]-hard. We describe the construction incrementally, starting with a simple idea and refining it repeatedly until a correct reduction is obtained. In this incremental process we will repeat the two steps of I) improving on the newest *version* of the reduction (thus creating a new version), and II) finding a way for the robots to *cheat* in the improved version, which proves that the newest version is still incorrect. We say informally that there is a way to cheat, if there is a schedule of length at most ℓ that is not in the class of intended solutions. We will describe the class of intended solutions later.

Formally the input is an instance $\mathcal{I}_{\text{MCC}} = (G = (V_1 \dot\cup V_2 \dots \dot\cup V_k, E), k)$ of Multi-Colored Clique (short: MCC), parameterized by the number of colours k where every $V_i = \{v_{i,1}, \dots, v_{i,|V_i|}\}$. The goal is to construct an instance $\mathcal{I}_{\text{GCMP1}} = ((G' = (V', E'), (\mathcal{M} = \{R_1\}, \mathcal{F}), \ell), \ell)$, parameterized by the budget ℓ, i.e. the total number of admissible moves. There is only one marked robot, R_1.

For the remainder of this chapter, we call a *full* (resp. *empty*) *path* a subgraph that is a path with (resp. without) robots. We say we *connect* two disjoint sets of vertices V_1 and V_2 with a full (or empty) path of length n, if we create a new path $P = (p_1, \dots, p_n)$ and connect (with an edge) p_1 to the vertices in V_1 and connect p_n to the vertices in V_2. The sets V_1 and V_2 are the *endpoints* of P.

3.1 Towards a working reduction

The general idea is to create a graph such that R_1 is forced to "pick" k *lanes* to travel through on his path to t_1. The lanes map bijectively onto vertices in G. We force i) R_1 to pick exactly one vertex from every colour in G, and ii) the existence of an edge between all picked vertices. Conditions i) and ii) together make for a clique in G.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

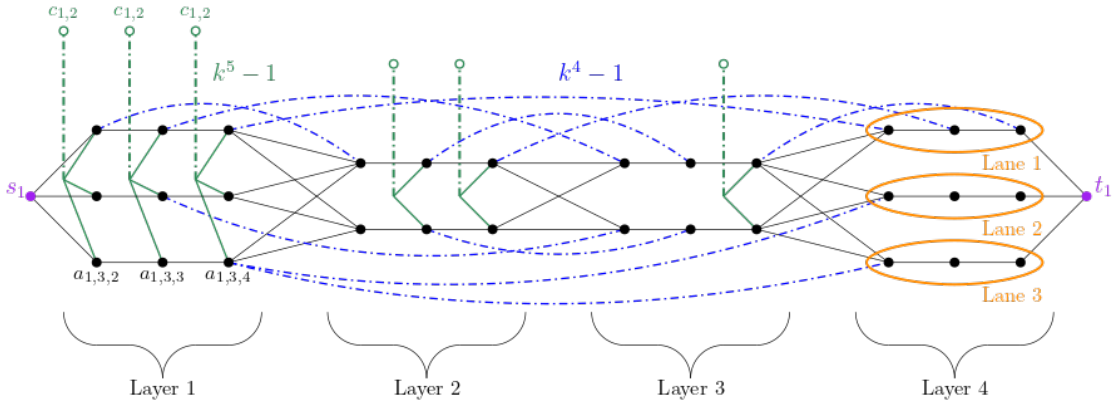

Figure 3.1: Version 0 of the reduction. The clear paths are green. The return paths are blue. Dot-dashed lines signify full paths.

Version 0. For the first version, consider Figure 3.1. First we set the budget $\ell = \binom{k}{2} \cdot k^5 + \binom{k}{2} \cdot k^4 + k \cdot (k-1) + 1$. For every vertex $v_{i,j} \in V_i$, we create a *lane*, a full path of $k-1$ vertices. For example, lane 3 in layer 1 corresponds to vertex $v_{1,3}$. For the sake of brevity we refer to the induced subgraph consisting of the vertices of the i-th layer as L_i. The layers are connected sequentially, while the lanes in each layer are connected in parallel, as shown in the figure. The reason the lanes have length $k-1$ is that they are used to check adjacencies between the vertex they represent and another picked vertex. We name the lane vertices as $a_{m,j,n}$ for $m, n \in [k], m \neq n, j \in [|V_m|]$. Moving forward we define $a_{m,j,n}$ in the following way:

1. if $m < n$, it is the n-th vertex in the j-th lane of L_m, and
2. if $m < n$, it is the $n-1$-st vertex in the j-th lane of L_m.

There are two more sets of paths in the figure. For all $m, n \in [k], m < n$, create a vertex $c_{m,n}$ and connect $\{a_{m,j,n} \mid j \in [|W_m|]\}$ to $\{c_{m,n}\}$ with a full path of length $k^5 - 1$. Those are the *clear paths* (green in Figure 3.1). Further, for all edges (v_{m,j_1}, v_{n,j_2}), connect $\{a_{m,j_1,n}\}$ to $\{a_{n,j_2,m}\}$ with a full path (blue in Figure 3.1) of length k^4 — the return paths.

We start by explaining the intended schedule, describing our idea for how the robots should move if $\mathcal{I}_{\text{MCC}}$ is a yes-instance. Assume there is a clique $X = \{v_{1,x_1}, \ldots, v_{k,x_k}\} \subseteq V$ in G. R_1 starts at s_1 and walks through the graph from left to right, picking in each layer the lane corresponding to the vertex in X; To pass layer L_m, R_1 picks lane x_m. Whenever R_1 is blocked by a robot R sitting on a vertex $a_{m,x_m,n}$, there are two possible cases:

1. $m < n$, which means $a_{m,x_m,n}$ is incident to a clear path: The robots inside the clear path as well as R shift towards $c_{m,n}$, leaving $a_{m,x_m,n}$ free.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

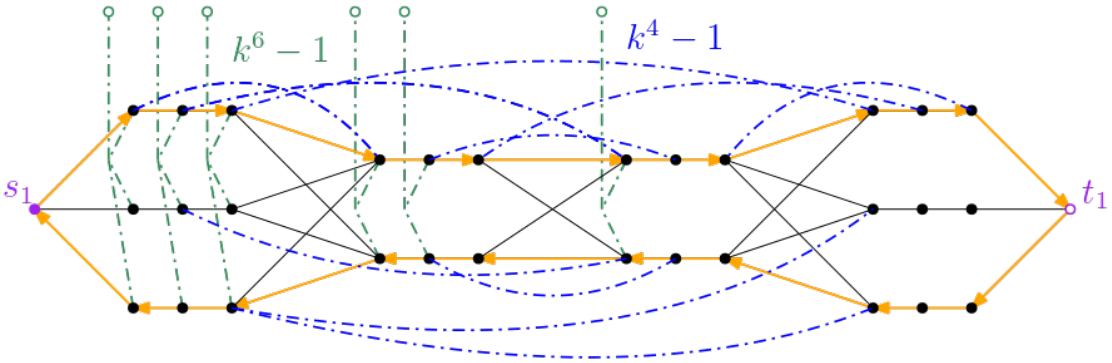

Figure 3.2: Moving around in the red cycle allows R_1 to cheat and get to t_1 without shifting any robots into clear paths.

2. $m > n$, which means there is a return path connecting $a_{m,x_m,n}$ to $a_{n,x_n,m}$. Since R_1 passed it earlier, $a_{n,x_n,m}$ is free now. The robots in the return path, as well as R shift over the path towards $a_{n,x_n,m}$, leaving $a_{m,x_m,n}$ free.

Note that in the second step, there is a return path leading back to a vertex previously passed by R_1. This is due to the fact that there is an edge between v_{m,x_m} and v_{n,x_n}. The general idea is that if the behaviour described above is forced in some way, then the existence of a feasible solution implies that X is a k-clique.

Secondly, note that there are different, equivalent schedules that also result in R_1 crossing to t_1 in ℓ moves. For example, instead of shifting robots into the clear paths once R_1 is standing before a corresponding blocker, an equivalent schedule may instead shift all the same blockers into clear paths right at the start before ever moving R_1. We consider two feasible schedules equivalent, if they differ only in the order of their moves.

The budget ℓ is picked to fit exactly the moves required by the intended solution. Let $\ell = \binom{k}{2}k^5 + \binom{k}{2} \cdot k^4 + k \cdot (k-1) + 1$. The first term covers the cost of shifting robots over clear paths towards C, the second term the cost of shifting over return paths and the rest covers the cost of moving R_1.

Cheat 0: The robots can go around in a cycle without ever shifting towards C.

There is a way for the robots to cheat such that within the same budget ℓ, there is a solution that is different to the intended solution. Figure 3.2 shows a directed cycle in our construction, along which R_1, as well as all robots present in the cycle in the initial configuration, can rotate. In total we move $1 + 2 \cdot (k-1) \cdot k$ robots, $2 + 2 \cdot (k-1) \cdot k$ times, requiring $(1 + 2 \cdot (k-1) \cdot k) \cdot (1 + 2 \cdot (k-1) \cdot k) = 4k^4 - 8k^3 + 10k^2 - 6k + 2$ moves. These moves clearly fit into the budget $\ell = \binom{k}{2} \cdot k^5 + \binom{k}{2} \cdot k^4 + k \cdot (k-1) + 1$ for large enough k and so the reduction is not yet correct.

Version 1. To prevent Cheat 0, we we introduce Version 1 of the construction. The goal is to prevent the robots from cheating by simply rotating around until R_1 sits on t_1.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

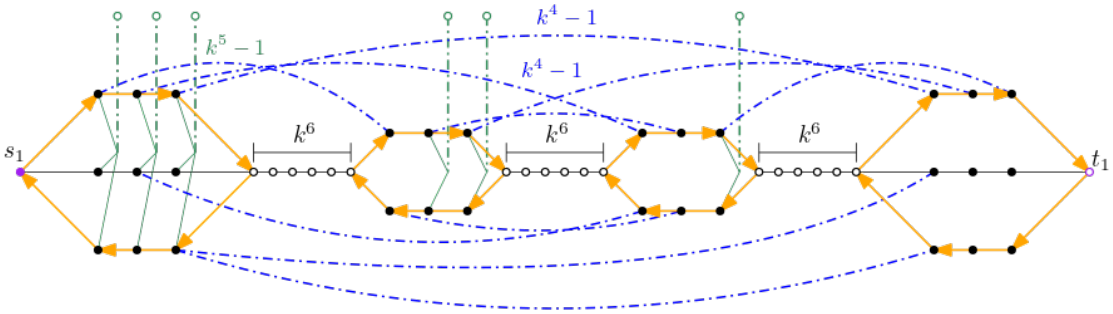

Figure 3.3: Version 1 after inserting separators in between successive layers.

We do this by adding *tiny separators* between two successive layers. The idea is to make passing through the separators so "expensive" that only R_1 can pass through it. The tiny separators have length k^6. The budget is extended to cover the cost of moving R_1 through the tiny separators and it is now $\ell = \binom{k}{2}k^5 + \binom{k}{2} \cdot k^4 + k \cdot (k-1) + (k-1) \cdot k^6 + 1$.

Cheat 1: The robots can rotate around all layers in G_D and the blue paths do not need to be used.

We run into a similar problem as before: The robots still do not need to vacate the lanes by shifting towards C. Instead they can now rotate inside of every layer, saving the cost of shifting over the clear paths. We must still find a way to force this behaviour.

Version 2. How do we prevent the robots to ignore the free vertices in C? We need to find a method to force vertices to be empty after R_1 passes through the main portion of the graph. For Version 2, we add new components to the graph in order to be able to force this behaviour. We refer henceforth to the part of the graph, where R_1 is intended to pick the lanes as the *decision component* G_D. In Version 1, the decision component G_D plus t_1 constituted the whole graph.

For the current version we add the *test component* G_T and the *separator component* G_S. The separator component separates the other two components (Figure 3.4). The separator component G_S is a long empty path of length k^{20}. Its purpose is to make it too expensive for any robot that is not R_1 to pass, as the budget will be picked to allow only one robot to pass through G_S. The test component G_T consists of a single full path. Every vertex in it has long full paths of length $k^{11} - 1$ leading back towards vertices in G_D. We call those long paths the *test paths*. The paths should be so long that if R_1 were to pass a test path — which would necessarily also involve moving out every robot inside it — the budget would be exceeded.

Since the only way for R_1 to reach t_1 is to pass through G_T, the blockers inside will need to vacate it. The only way for them to do this is by shifting over the test paths towards G_D. In order to avoid the blockers moving towards other vertices in G_T and shifting over test paths other than the intended ones, we subdivide every edge in G_T, k^8 times. We use the mechanism of test paths to force more behaviour for the robots in G_D.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

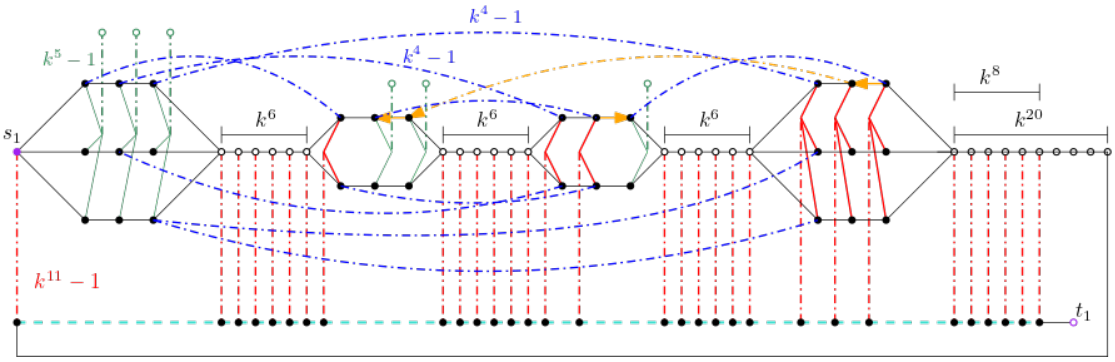

Figure 3.4: Version 2. We add the test paths and the separator between the main part and the test part. Dot dashed lines signify full paths and the dashed line in the test component stands for the fact that each edge is subdivided k^8 times.

We add test paths connecting G_T and $L \cup Q \cup G_S$ as shown in Figure 3.4. The budget is now $\ell = \binom{k}{2}k^5 + \binom{k}{2} \cdot k^4 + ((\binom{k}{2} + (k-1) \cdot k^6 + k^8) \cdot k^{11}) + k \cdot (k-1) + (k-1) \cdot k^6 + 1$. The third term in ℓ covers the cost of shifting over all return paths.

The test paths now force three force...

1. all vertices in the tiny separators,
2. at least one vertex in each $\{L_{m,i,n} \mid i \in |W_i|\}$ for all $1 \leq n < m \leq k$, and
3. the first k^8 vertices in $G_{S,1}$

...to be empty when R_1 goes to pass G_T.

Cheat 2: The robots need not shift back over $\binom{k}{2}$ return paths. In layers $L_2, \ldots, L_{k-1}$, the robots intended to move back over a return path can instead find homes in the vertices in the same layer that are vacated by shifting into a clearpath, saving the cost of shifting over return paths.

In Figure 3.4, the robot sitting on $a_{3,1,2}$ was intended to shift towards $a_{2,1,3}$ over the return path, but it instead decides to occupy the vertex one move to the right. The other robots compensate for the hole created in Layer 2 and the excess robot in Layer 3 as shown in orange in the Figure.

Version 3. The problem with the previous version is that some robots in L_m for some $m \in [k]$ may decide to not move to a different layer over a return path. Instead they may move towards an open space inside the same layer. When such a robot in L_m ($m \in [k]$) decides to not shift towards layer L_m, it creates a free space in L_m and an excess robot in L_m that needs to be dealt with. In previous versions we restricted the movement of blockers by adding separators. While this approach may work again, we decide on a more elegant solution. We change the lengths of the return paths so that a return path

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

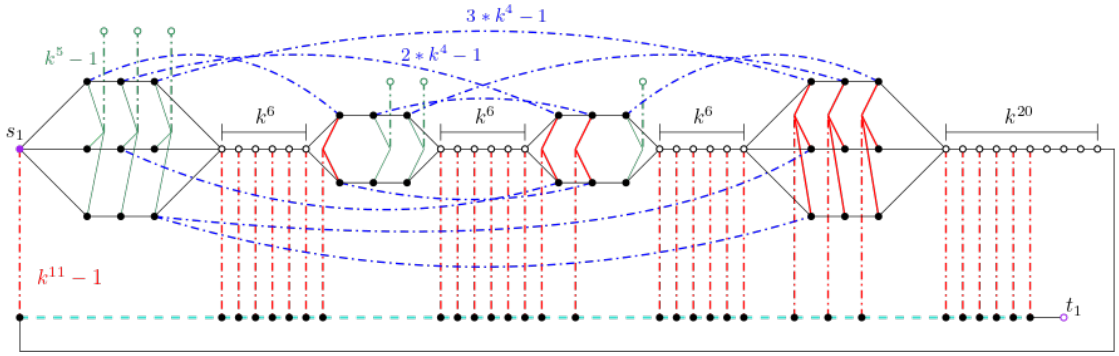

Figure 3.5: Version 3. After varying the lengths of the return paths, the budget must be used up precisely if one wants to achieve a feasible solution. This makes it impossible for robots to make use of Cheat 2, as there would be no budget leftover for intra-layer movement of blockers.

from $a_{m,x_m,n}$ to $a_{n,x_n,m}$ (assuming $m < n$) has length $(m - n) \cdot k^4 - 1$. By making this change, the cheat in Figure 3.4 is no longer possible, as the long return path shift from $a_{4,1,2}$ to $a_{2,1,4}$ now costs twice as much.

Version 3 is the final, correct version of our construction and we describe it formally in the following section before proving its correctness.

3.1.1 Conclusion

We have given an example of an incremental design procedure that could be similar to the thought process of a researcher trying to devise a valid reduction for the purpose of a hardness proof for a new, interesting problem. It starts with a simple idea for how a feasible solution is intended to look like. This idea is tested multiple times when we find ways to cheat by finding other, simpler solutions that stay within the budget, which make the proof fall apart. After multiple targeted revisions, we fail to find another cheat and are left with a correct — albeit more complex — reduction.

3.2 A Formal Hardness Proof

The previous section described the incremental process of how one could arrive at a reduction that seems complex at first glance. But our job is not finished until the idea is formalized and its correctness proven. That will be the goal of this section.

We receive an instance $\mathcal{I}_{\text{MCC}} = (G = (V_1 \dot\cup V_2 \dots \dot\cup V_k, E), k)$ of MCC, parameterized by the number of colours k where every $V_i = \{v_{i,1}, \dots, v_{i,|V_i|}\}$. The goal is to construct an instance $\mathcal{I}_{\text{GCMP1}} = ((G' = (V', E'), (\mathcal{M} = \{R_1\}, \mathcal{F}), \ell), \ell)$, parameterized by the budget ℓ, i.e. the total number of admissible moves. For a correct parameterized reduction three conditions described must be fulfilled:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

1. $\mathcal{I}_{\text{MCC}}$ is a yes-instance if and only if $\mathcal{I}_{\text{GCMP1}}$ is a yes-instance.
2. $\ell < g(k)$ for some computable function g.
3. The reduction algorithm runs in FPT time, i.e. in $f(k) \cdot |G|^{O(1)}$ for some computable function f.

3.2.1 Construction

We start with a formal description of the construction, i.e. G'. The graph consists of three components: i) *decision component* G_D, ii) the *separator component* G_S, and iii) the *test component* G_T.

We start constructing G_D by creating a *lane* (= a full path) of length $k-1$ for every vertex in $v_{m,i} \in V$, and let $L_{m,i}$ be the set of vertices of that lane. The individual vertices in $L_{m,i}$ are indexed as $a_{m,i,n}$ with $n \in [k] \setminus \{m\}$. Let $L_m = \bigcup_{i \in [|V_m|]} L_{m,i}$ be the (vertices of the) m-th layer and let $L = G[V(\bigcup_{m \in [k]} L_m)]$. Next create a tiny separator (an empty path) with vertices $Q_i = \{q_{m,1}, \ldots, q_{m,k^6}\}$ for each $m \in [k-1]$. Further, let $Q = \bigcup_{m \in [k-1]} Q_i$. We connect all layers in series using the tiny separators, as follows: Connect the following vertices with an edge:

1. each $a_{m,i,k}$ with $q_{m,1}$ for each $m \in [k-1]$.
2. each $a_{m,i,1}$ with q_{m,k^6} for each $m \in [k] \setminus \{1\}$.
3. s_1 with each $a_{1,i,2}$, with $i \in [|V_1|]$.

Create for each $1 \leq m < n \leq k$ a vertex $c_{m,n}$. Connect each $c_{m,n}$ and $L_{m,i}$ with a clear path $C_{m,n}$, i.e. a full path of length $k^5 - 1$. We call $C = \bigcup_{1 \leq m < n \leq k} \{c_{m,n}\} \cup C_{m,n}$.

For each edge $(v_{m,i}, v_{n,j}) \in E$ with $m < n$, connect $a_{m,i,n}$ and $a_{n,j,m}$ with a return path $R_{m,n}$ — a full path of length $(m-n) \cdot k^4 - 1$. Let $R = \bigcup_{1 \leq m < n \leq k} R_{m,n}$.

The decision component $G_D = G[\{s_1\} \cup L \cup Q \cup C \cup R]$ is the graph induced by the vertex sets defined above.

The separator component $G_S = \{b_1, \ldots, b_{k^{20}}\}$ is an empty path. It connects the last layer of G_D to the first vertex in the test component. Let also the first k^8 vertices of the path be called $G_{S,1}$ and the rest of the path $G_{S,2}$.

For the test component G_T, we start with a full path P of length $\binom{k}{2} + k^6 \cdot (k-1) + k^8 + 1$. Connect each vertex in P with one or more vertices in $V(G_D) \cup V(G_{S,1})$, in the following way: We define a finite sequence D of vertex sets from $V(G_D) \cup V(G_{S,1})$. Then we connect the n-th vertex in P and the n-th element in D with a test path, a full path of length $k^{11} - 1$. We define D as the **flattened** sequence of sequences $((d_{1,n})_{n=2}^{1}, (d^*_{1,n})_{1 \leq n \leq k^6}, (d_{2,n})_{n=2}^{2}, \ldots, (d^*_{k-1,n})_{1 \leq n \leq k^6}, (d_{k,n})_{n=2}^{k}, (d^*_{k,n})_{1 \leq n \leq k^8})$. The first element $(d_{1,n})_{n=2}^{1}$ is technically empty, and is listed only for the sake of uniformity. Then the single elements are defined as:

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

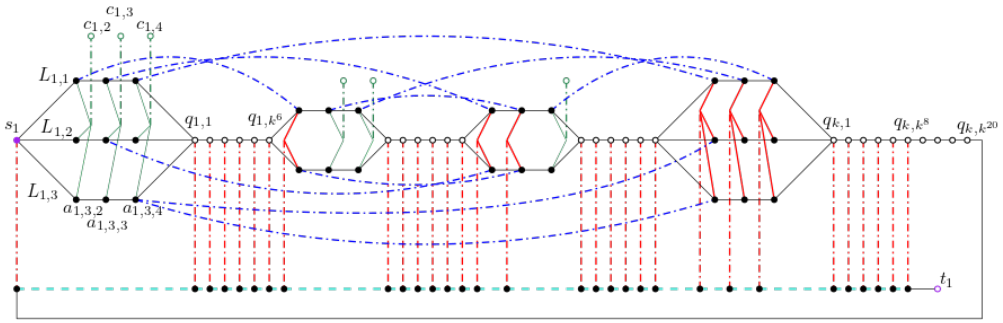

Figure 3.6: The whole construction of G' along with some labels to clarify the naming conventions.

1. each $d_{m,n} = \{a_{m,i,n} \mid i \in [|V_m|]\}$,
2. for $m \in [k-1]$, each $d^*_{m,n} = \{q_{m,n}\}$, and
3. $d^*_{k,n} = \{b_n\}$

Finally, every edge in P is subdivided $k^8 - 1$ times, leaving the newly created vertices free. The resulting path is the test component G_T.

Budget	Value	Use
$\ell_{1,1}$	$k(k-1) + k^6(k-1) + 1$	Move R_1 from s_1 onto b_1.
$\ell_{1,2}$	$k^{20} + (k^8+1)(k^6(k-1) + \binom{k}{2} + k^8) + 2$	Move R_1 from b_1 onto t_1.
ℓ_2	$k^5\binom{k}{2}$	Shift over clear paths.
ℓ_3	$\frac{1}{6}k^5(k^2-1)$	Shift over return paths.
ℓ_4	$k^{11}(k^6(k-1) + \binom{k}{2} + k^8 + 1)$	Shift over test paths.

Table 3.1: The partitioning of the budget and the parts' respective intended uses.

Figure 3.6 shows the whole construction, along with some labels in order to clarify the naming conventions. Finally, we set the budget to $\ell = \ell_{1,1} + \ell_{1,2} + \ell_2 + \ell_3 + \ell_4$. The different parts of the budget and their intended uses are outlined in Table 3.2.1. This completes the formal description of the construction. The next is proving correctness.

3.2.2 Proof of correctness

1. $\mathcal{I}_{\text{MCC}}$ is a yes-instance if and only if $\mathcal{I}_{\text{GCMP1}}$ is a yes-instance.
2. $\ell < g(k)$ for some computable function g.
3. The reduction algorithm runs in FPT time, i.e. in $f(k) \cdot |G|^{O(1)}$ for some computable function f.

2. and 3. are clearly true. To prove 1., we prove the forward- and the backward direction separately.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Proof of Forward Direction "⇒". We show the forward direction by assuming an arbitrary yes-instance for the parameterized MCC problem and defining a schedule for the corresponding GCMP1 instance, parameterized by ℓ. In the new instance R_1 reaches t_1 and the total energy expended does not exceed ℓ. Let $\mathcal{I}_{\text{MCC}} = ((V_1 \dot\cup V_2 \dots V_k, E), k)$ be such an arbitrary yes-instance and let $X = \{v_{1,x_1}, v_{2,x_2}, \dots, v_{k,x_k}\}$ be a k-clique in G.

We define the desired schedule by executing the following steps in sequence:

1. **Shift over clear paths.** For each $1 \leq m < n \leq k$, shift the blocker sitting on $a_{m,x_m,n}$ over the incident clear path towards $c_{m,n}$. This requires $\ell_2 = \binom{k}{2} \cdot k^5$ moves.

2. **Shift over return paths and move R_1 from s_1 onto b_1.** Move R_1 towards b_1, while picking the correct lanes: In layer m, the robot picks the lane x_m. At some point the robot cannot advance further due to a blocker sitting in its path. Let $a_{n,x_n,m}$ be the blocked vertex. In this case we have that $m < n$. Further, due to the fact that X is a clique, there is a return path leading from $a_{n,x_n,m}$ towards $a_{m,x_m,n}$. Shift the vertices in the return path, along with the robot that is blocking R_1, towards $a_{m,x_m,n}$. Our main robot may advance again. The total cost of moving R_1 requires $\ell_{1,1} = k(k-1) + k^6(k-1) + 1$ moves and shifting over return paths requires $\ell_3 = \sum_{1 \leq m < n \leq k} k^4 \cdot (n-m) = \frac{1}{6}k^5(k^2-1)$ moves.

3. **Shift over test paths.** At this point in time, every robot in G_T is incident to a test path with its endpoint in $G_D \cup G_{S,1}$ containing exactly one free vertex. Indeed, for the test paths with an endpoint $\{a_{m,i,n} \mid i \in [|V_m|]\} \subset V(L)$ for some $1 \leq m < n \leq k$, the free vertex is $a_{m,x_m,n}$ — the vertex that R_1 passed on his way through G_D. Clearing G_T requires $\ell_4 = k^{11}(k^6(k-1) + \binom{k}{2} + k^8 + 1)$ moves.

4. **Move R_1 from b_1 onto t_1.** This requires $\ell_{1,2} = k^{20} + (k^8+1)(k^6(k-1) + \binom{k}{2} + k^8) + 2$ moves.

The total number of moves expended is $\ell = \ell_{1,1} + \ell_{1,2} + \ell_2 + \ell_3 + \ell_4$, which proves the forward direction. □

Proof of Backward Direction "⇐". Partition the budget into the five parts listed in Table 3.2.1. We will show that in any feasible solution, every "Use" in the table is both necessary for feasibility and requires at least the number of moves listed in the "Value" column. Both of these facts together imply that there is no budget left to allow for other, unpredicted moves by the robots. Then a simple argument finishes the proof.

1. **Use: "Shift over test paths".** We start with the claim that it is not possible for R_1 to reach G_T by crossing a test path. It requires at least $2 * \sum_{i=1}^{\frac{k^{11}}{2}} i = \frac{k^{22}}{8} + \frac{k^{11}}{4}$ moves to allow R_1 to pass a test path, as every blocker inside the path would need to move out as well. For large enough k this is not within the budget however, as the dominating term in the budget ℓ is k^{20}. Therefore the whole test component

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

needs to be cleared in order to let R_1 pass through to t_1. A blocker that needs to make space cannot pass through the whole G_S, as this requires k^{20} moves, but R_1 needs this budget to pass through G_S itself. Therefore all blockers must move out by shifting over the test paths towards $L \cup Q \cup G_{S,1}$. The shifting requires k^{11} moves per test path, i.e. $\ell_4 = k^{11}(k^6(k-1) + \binom{k}{2} + k^8 + 1)$ moves in total. Going forward, we subtract ℓ_4 from the available budget for other moves, as ℓ_4 moves are reserved for this specific use.

2. **Use: "Move R_1 from b_1 onto t_1".** Once R_1 reaches b_1, it needs to cross G_S and G_T in order to reach t_1. This clearly requires $|V(G_S) \cup V(G_T)| + 1$ moves, which is exactly $\ell_{1,2}$. We reserve $\ell_{1,2}$ moves for this purpose.

3. **Use: "Shift over clear paths".** Consider the number of robots that need to shift into $L \cup Q \cup G_{S,1}$. In the initial configuration, there are $\binom{k}{2}$ blockers too many in those vertices, meaning that those blockers need to find a home somewhere else. Moving into $G_{S,2}$ would require k^8 moves, which is not within the remaining budget. Therefore those robots need to shift over clear paths towards the free vertices in C. Therefore $\ell_2 = k^5\binom{k}{2}$ moves are reserved for this purpose.

4. **Use: "Move R_1 from s_1 onto b_1".** Using similar reasoning as before, we claim that R_1 cannot pass through a return path to get closer to b_1, as this would require at least $(k^4)^2 = k^8$ moves, which is not in the remaining budget. Therefore R_1 needs to move over L and Q. It is easy to see that the length of any shortest path from s_1 to b_1 using only vertices in $L \cup Q$ is exactly $\ell_{1,1}$, which is a trivial lower bound for this use.

5. **Use: "Shift over return paths".** The remaining budget is $\ell_3 = \frac{1}{6}k^5(k^2-1)$. Note first that every blocker in G_T necessarily shifts exactly over the test path incident to its initial position, and not some other test path. This is because the minimum distance between two different such blockers is k^8 and moving to the endpoint of a different test path in G_T would exceed the remaining budget.

   Let $X_A \subseteq L$ be the vertices ("holes") that are freed up when all $\binom{k}{2}$ excess blockers shift towards $\{c_{m,n} \mid 1 \leq m < n \leq k\}$. Let $X_B \subseteq L$ be the vertices towards which blockers coming from G_T shift. For the sake of brevity we define $X_A^m = X_A \cap L_m$ and $X_B^m = X_B \cap L_m$. The blockers that are in X_B in the initial configuration must either move directly into X_A or shift towards a vertex in X_A in order to potentially find a feasible solution. Let $f_A(m) = |X_A^m|$ and $f_B(m) = |X_B^m|$. Considering the fact that f_A increases monotonically and f_B decreases monotonically, we say that generally the blockers need to move from higher to lower layers (from right to left in Figure 3.6). We aim to compute the minimum number N of layers that the blockers need to move back.

   In order to simplify the argument, assume that blockers in layer m first move to free holes inside the same layer, before looking to other holes in different layers. Let $f(m) = f_B(m) - f_A(m)$ be the number of excess robots (when $f(m) > 0$) or

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the number of free holes (when $f(m) < 0$) after this process finishes. Due to the construction, $f(m) = (m-1) - (k-m) = 2m - k - 1$. Observe that $f(m) \leq 0$ for $m \leq \frac{k+1}{2}$ and $f(m) \geq 0$ for $m \geq \frac{k+1}{2}$. This confirms the above claim that blockers in general need to move from higher to lower layers.

Let the function $g : [k-1] \to \mathbb{N}$ define the minimum number of blockers that need to cross over layer $m+1$ into layer m or over layer m into a layer $n < m$. Since f is monotonically increasing in the range $[1, \lfloor \frac{k+1}{2} \rfloor]$, we can define for the same range $g(m) = \sum_{i=1}^{m} -f(i)$ as the number of holes in the first m layers. Similarly, for $m \in [\lfloor \frac{k+1}{2} \rfloor + 1, k-1]$, the expression $g(m)$ is simply the total number of excess robots in layers $m+1$ through k, so $g(m) = \sum_{i=m+1}^{k}$. We note that the two sums are both equal to the expression $km - m^2$, and so $g(m) = km - m^2$ for all $m \in [k]$. Therefore we can deduce $N = \sum_{m=1}^{k} g(m) = \frac{1}{6}k(k^2-1)$.

Observe that moving a robot from layer m to layer n (w.l.o.g. $m < n$) over Q requires at least $(n-m) \cdot k^6$ moves, that is the length of the tiny separators. Instead shifting from layer n to layer m requires at least $(n-m) \cdot k^4$ moves. The minimum cost of moving all robots to the right layer is therefore at least $N \cdot k^4 = \frac{1}{6}k^5(k^2-1) = \ell_3$ moves.

We have shown that all uses of the budget are necessary for a feasible solution and they each require at least the number of moves they are attributed in Table 3.2.1. Therefore robots cannot do any other moves not specified in the table, and especially there may be no intra-layer movement of blockers. It is left to prove that if there exists a feasible schedule, then G contains a k-clique.

Let S be a feasible schedule for $\mathcal{I}_{\text{GCMP1}}$. Let $\{L_{1,i_1}, \ldots, L_{k,i_k}\}$ be the lanes that R_1 picks on its way through G_D. Assume towards a contradiction that $\{v_{1,i_1}, \ldots, v_{k,i_k}\}$ is not a k-clique, due to the non-existence of the edge (v_{m,x_m}, v_{n,x_n}) with $m < n$. We know that at some point, R_1 needs to pass vertex $a_{n,x_n,m}$ and is blocked. However, there is no return path leading towards $a_{m,x_m,n}$. Furthermore, any other potential return paths lead towards occupied vertices. In any way, one would need to move at least one blocker in L around over non-return paths in order to vacate $v_{n,x_n,m}$, which was shown to be prohibited in feasible schedules. Contradiction.

□

The correctness of the reduction implies the main Theorem for this chapter:

Theorem 1. GCMP1 *is W[1]-hard when parameterized by* ℓ.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 4

Treedepth to the Rescue

In the previous chapter we showed that GCMP is difficult to solve, even when the number of marked robots as well as the energy budget are assumed to be small. In fact, the problem is already W[1]-hard when there is even only one marked robot. We next remove ℓ from the parameter and replace it with the treedepth of the input graph. The idea is that if the graph is simple or structured in some sense, this could potentially restrict the solution space for possible schedules so much that the problem becomes tractable. This chapter aims to provide a positive result in contrast to the negative result from the previous chapter. This result is written up in the main theorem in this chapter:

Theorem 3. GCMP *is fixed-parameter tractable when parameterized by the number* $k = |\mathcal{M}|$ *of marked robots plus the treedepth of the input graph.*

The proof is structured as one large case distinction, differentiating between whether or not the number of free vertices in G exceeds a function of the treedepth k. A different proof is then provided for each case.

4.0.1 Case 1: Few Free Vertices

.

The main goal of this section is to prove the following lemma:

Lemma 1. *Let* $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \ell)$ *be an instance of* GCMP *where* G *has treedepth at most* td *and there are at most* n_f *free vertices. If there is a schedule for* $\mathcal{I}$*, then there is an optimal schedule that uses energy at most* $\gamma(n_f, k, \mathsf{td})$ *for some computable function* γ*. Moreover, an optimal schedule (if it exists) can be computed in time* $\gamma(n_f, k, \mathsf{td}) \cdot n^{\mathcal{O}(1)}$*.*

The fact that the above lemma is true also means that n_f is bounded by some function $g(k, td)$. This implies that GCMP is FPT, as with that all parameters for γ are effectively

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

fixed. For all lemmas in this subsection, presume that we are given an instance $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \ell)$. The first k robots in $\mathcal{R}$ are the marked robots, i.e. $\{R_i \mid i \in [k]\} = \mathcal{M}$. Let S be a schedule for $\mathcal{I}$. We call a vertex set $X \subseteq V(G)$ *fully blocked* at a time step s, if every vertex in X is occupied by a blocker from $\mathcal{F}$ at that time step.

Definition 2 (Configurations)**.** *A* pseudo-configuration *for $\mathcal{I}$ is a pair (τ, Q) where $\tau = (\tau[1], \ldots, \tau[k])$ is a tuple of vertices and $Q \subseteq V(G)$ is a vertex set. Let $V(\tau) = \cup_{i \in k}\{\tau[i]\}$. A* configuration *for $\mathcal{I}$ is a pseudo-configuration (τ, Q) where (i) $|V(\tau)| = k$, (ii) $|Q| = |\mathcal{F}|$ and (iii) $V(\tau) \cap Q = \emptyset$. We say that a configuration (τ, Q) is* the starting configuration *if for each $i \in [k]$, $\tau[i]$ is the starting vertex of R_i and Q is the set of starting vertices of the robots in $\mathcal{F}$. We say that a configuration (τ, Q) is* a destination configuration *if for each $i \in [k]$, $\tau[i]$ is the destination vertex of the robot R_i.*

Configurations describe the state of G, including the positions of the marked and unmarked robots, at some time step in a schedule. The unmarked robots are all identical and so it is not required to differentiate between them. Instead it is sufficient to keep track of only the set of vertices occupied by unmarked robots. The next three definitions follow this intuition:

Definition 3 (Moves between configurations)**.** *We say that there is a* move *from a configuration (τ_1, Q_1) to a configuration (τ_2, Q_2) if:*

- *either $Q_1 = Q_2$ and τ_1 and τ_2 differ at exactly one index $i \in [k]$ and $\tau_1[i]\tau_2[i] \in E(G)$; or*
- *$\tau_1 = \tau_2$ and $Q_1 \Delta Q_2$[1] has exactly two vertices u, v and $uv \in E(G)$.*

Definition 4 (Induced configurations)**.** *Given a schedule S for the instance $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \ell)$, we define the configuration* induced *by S at each time step s in the natural way, that is, it is the tuple (τ_s, Q_s) where $\tau_s[i]$ is the vertex occupied by robot $R_i \in \mathcal{M}$ at time step s and the robots in $\mathcal{F}$ occupy exactly the vertices in Q_s at the same time step.*

Definition 5 (Legal sequences)**.** *A sequence $(\tau_0, Q_0), \ldots, (\tau_t, Q_t)$ of configurations is called* legal *if: (i) (τ_0, Q_0) is the starting configuration; and (ii) (τ_t, Q_t) is a destination configuration; and (iii) for every $i \in [t-1]_0$, there is a move from (τ_i, Q_i) to (τ_{i+1}, Q_{i+1}).*

In simple words, a sequence of configurations is considered legal if and only if it represents a valid solution, i.e. a schedule for some initial configuration set by an instance $\mathcal{I}$ by which each marked robot ends on its destination. Observe that any induced configuration of a solution schedule is therefore automatically a legal sequence. We say that a legal sequence of configurations is *optimal* if its length is one plus the number of time steps in an optimal schedule.

[1] The symmetric difference Δ is defined as: $Q_1 \Delta Q_2 = (Q_1 \setminus Q_2) \cup (Q_2 \setminus Q_1)$.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Definition 6. *Consider two disjoint vertex sets Z_1 and Z_2 and a bijection $\phi : Z_1 \to Z_2$. The bijection $\phi^\star : V(G) \to V(G)$ is defined as follows. For every $v \in V(G)$,*

- *if $v \in Z_1$, then $\phi^\star(v) = \phi(v)$;*
- *if $v \in Z_2$, then $\phi^\star(v) = \phi^{-1}(v)$; and*
- *otherwise, $\phi^\star(v) = v$.*

The map $\phi^\star$ emulates ϕ for all vertices except for those in $Z_1 \cup Z_2$, for which it instead returns the (pre-)image of the argument. This function will be used as a tool to show that certain vertices can be replaced by others in a schedule while retaining validity, implying that those vertices are redundant. For this purpose we define the process of applying $\phi^\star$ to an existing schedule:

Definition 7. *Consider two disjoint vertex sets Z_1 and Z_2 and a bijection $\phi : Z_1 \to Z_2$. The operation of* applying the bijection ϕ to Γ from time step s onwards *involves defining a new sequence $\hat{\Gamma} = (\hat{\tau}_0, \hat{Q}_0), \ldots, (\hat{\tau}_t, \hat{Q}_t)$ of pseudo-configurations as follows.*

1. *For every $i < s$, set $(\hat{\tau}_i, \hat{Q}_i) := (\tau_i, Q_i)$.*
2. *For every $i \geq s$,*
   a) *set $\hat{Q}_i := \cup_{v \in Q_i} \phi^\star(v)$; and*
   b) *for every $j \in [k]$, set $\hat{\tau}_i[j] := \phi^\star(\tau_i[j])$.*

We finally define the notion of *strong isomorphism* between connected components of a graph. The definition effectively formalizes that two components are "equal" (or interchangable) with respect to how robots inside might move around and with respect to how they can interact with the rest of the graph.

Definition 8. *Let $Z \subseteq V(G)$. We say that a pair of connected components C_1 and C_2 of $G - Z$ are* strongly isomorphic *with respect to Z if there is a bijection $\phi : V(C_1) \to V(C_2)$ such that ϕ is a graph isomorphism from C_1 to C_2, and moreover, for every $u \in V(C_1)$ and $v \in Z$, $uv \in E(G)$ if and only if $\phi(u)v \in E(G)$. We drop the explicit reference to Z if it is clear from the context. We also say that ϕ is a* witness *for C_1 and C_2 being strongly isomorphic.*

Lemma 2. *Let $Z \subseteq V(G)$ and consider a pair of connected components C_1 and C_2 of $G - Z$ that are disjoint from the set of terminals of $\mathcal{M}$. Suppose that C_1 and C_2 are strongly isomorphic with respect to Z, witnessed by ϕ. Suppose also that at time step c, $V(C_1)$ and $V(C_2)$ are fully blocked in S. Let $\hat{\Gamma} = (\hat{\tau}_0, \hat{Q}_0), \ldots, (\hat{\tau}_t, \hat{Q}_t)$ denote the sequence of pseudo-configurations obtained by applying the bijection ϕ to Γ from time step $c > 0$ onward. Then, the following hold:*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

1. *The length of* $\hat{\Gamma}$ *is the same as that of* Γ.

2. *The sequence* $\hat{\Gamma}$ *is a legal sequence of configurations.*

Proof. The length of $\hat{\Gamma}$ is clearly the same as that of Γ, based on the construction.

To prove 2., we start by proving that $\hat{\Gamma}$ is a sequence of configurations. If it was not a sequence of configurations, there would be an $s \in [c, k]$, such that either (i) $|V(\hat{\tau}_s)| \neq k$, or (ii) $|\hat{Q}_s| \neq |\mathcal{F}|$, or (iii) $V(\hat{\tau}_s) \cap \hat{Q}_s \neq \emptyset$. For (i) to be true would require a pair of duplicates in $(\hat{\tau}_s[1], \ldots, \hat{\tau}_s[k])$. This cannot occur, because there are no duplicates in $(\tau_s[1], \ldots, \tau_s[k])$ and ϕ is a bijection. Likewise, (ii) can also not be true. We need to show that $V(\tau_s) \cap Q_s = \emptyset$. We know that $V(\tau_s) \cap Q_s = \emptyset$. Since $\phi^\star$ is injective, it is not possible that a vertex would be mapped to twice.

Next we show that the sequence of configurations is legal. We prove, according to Definition 5:

1. $(\hat{\tau}_0, \hat{Q}_0)$ **is the starting configuration**:

   Since $c > 0$, we have that $(\tau_0, Q_0) = (\hat{\tau}_0, \hat{Q}_0)$.

2. $(\hat{\tau}_t, \hat{Q}_t)$ **is a destination configuration**:

   By definition $C_1 \cup C_2$ is disjoint from $\mathcal{M}$. It follows that for every $i \in [k] : \hat{\tau}_t[i] = \tau_t[i]$. Since (τ_t, Q_t) itself is a destination configuration, so is $(\hat{\tau}_t, \hat{Q}_t)$.

3. **For every** $i \in [t-1]_0$**, there is a move from** $(\hat{\tau}_i, \hat{Q}_i)$ **to** $(\hat{\tau}_{i+1}, \hat{Q}_{i+1})$:

   Assume towards a contradiction that there are time steps $\{i_m \mid \in [c, t-1]\}$ for which there is no move from $(\hat{\tau}_{i_m}, \hat{Q}_{i_m})$ to $(\hat{\tau}_{i_m+1}, \hat{Q}_{i_m+1})$. Let i^* be the first of these time steps. Consider the move (τ_{i^*}, Q_{i^*}) to $(\tau_{i^*+1}, Q_{i^*+1})$. We aim to prove that the move from $(\hat{\tau}_{i^*}, \hat{Q}_{i^*})$ to $(\hat{\tau}_{i^*+1}, \hat{Q}_{i^*+1})$ exists. Let (v_1, v_2) be the edge along which a robot moves in that timestep in Γ. Firstly observe that due to fact that ϕ is a witness of the strong isomorphism between C_1 and C_2 with respect to Z, the edge $\phi^\star(v_1), \phi^\star(v_2)$ exists. Therefore the only way for a move from $(\hat{\tau}_i, \hat{Q}_i)$ to $(\hat{\tau}_{i+1}, \hat{Q}_{i+1})$ to not exist, is if at timestep i^*, (i) $\phi^* star(v_1)$ is unoccupied, or (ii) $\phi^* star(v_2)$ is occupied. Notice that v_1 is occupied at time step i^* in *Gamma* and v_1 is unoccupied at timestep i^* in *Gamma*. Since $\phi^\star$ is a bijection, we have that after the first $i^* - 1$ moves are executed in $\hat{\Gamma}$, $\phi^\star(v_1)$ must also be unoccupied and $\phi^\star(v_2)$ must be occupied. Therefore the move from (τ_{i^*}, Q_{i^*}) to $(\hat{\tau}_{i^*}, \hat{Q}_{i^*})$ exists. Contradiction.

□

Lemma 3. *Let* $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \ell)$ *be an instance of* GCMP *with at most* n_f *free vertices. Consider a set* Z *in* G *and a set* $\mathcal{C} = \{C_1, \ldots, C_r\}$ *of connected components of* $G - Z$ *such that:*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

- *they are pairwise strongly isomorphic with respect to Z;*
- *$r > n_f + 3k + 1$; and*
- *C_r is fully blocked at time step 0 and disjoint from the terminals of $\mathcal{M}$.*

If there is a schedule for $\mathcal{I}$, then there is an optimal schedule in which $V(C_r)$ is fully blocked at every time step.

Proof. Assume that $\mathcal{I}$ is a yes-instance and pick a schedule S in a way such that (i) it is optimal, and (ii) the first time step s when C_r is no longer fully blocked is maximized. Notice that due to the fact that $r > n_f + 3k + 1$, there must be at least one other component C_i that is fully blocked at time step $s - 1$ (pigeon hole principle). Making use of the strong isomorphism between C_r and C_i w.r.t. Z, we define the sequence $\hat{\Gamma}$ of pseude-configurations by applying ϕ to Γ from time step $s - 1$ onwards. By Lemma 3, $\hat{\Gamma}$ is a legal sequence of configurations. Let $\hat{S}$ be the schedule corresponding to $\hat{\Gamma}$. Since $|\Gamma| = |\hat{\Gamma}|$, the two schedules S and $\hat{S}$ require the same amount of energy. Therefore $\hat{S}$ is also optimal. It breaks however our assumption that we picked S the be the schedule where C_r remains fully blocked for the longest possible time. Contradiction. □

From this lemma we can derive a useful corollary, that will be used extensively in the upcoming proofs.

Corollary 1. *Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \ell)$ be an instance of* GCMP. *Suppose there is a vertex v that remains occupied at every time step of an optimal schedule by a robot R. Then the energy used by an optimal schedule for $\mathcal{I}' = (G - v, \mathcal{R} = (\mathcal{M}, \mathcal{F} \setminus \{R\}), \ell)$ is the same as the energy used by an optimal schedule for $\mathcal{I}$. Moreover, given an optimal schedule for $\mathcal{I}'$, one for $\mathcal{I}$ can be produced in polynomial time.*

Lemma 4. *There are computable functions $\mu_1, \mu_2 : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and an algorithm that, given a connected graph G, a tree embedding (F, f) of G of depth at most d and a number $\eta \in \mathbb{N}$, runs in time $\mu_1(\eta, d) \cdot n^{\mathcal{O}(1)}$ and if G has more than $\mu_2(\eta, d)$ vertices, then it produces a set $Z \subseteq V(G)$ and a set $\mathcal{C} = \{C_1, \ldots, C_\eta\}$ of components of $G - Z$ that are pairwise strongly isomorphic with respect to Z.*

Let r be the root of F. For every $v \in V(F)$, let F_v be the subtree of F rooted in v. We call $V(F_v) \subseteq V(G)$ the set of pre-images of vertices in F_v. We call $s(i)$ an upper bound on the number of vertices rooted at any vertex at depth i in F, and similarly, $\delta(i)$ an upper bound on the number of immediate children of any vertex at depth i. Finally, we define a function $\chi(\eta, d, i) = \eta * s(i-1) * 2^{d-i} * 2^{s(i-1)^2 + (d-i) * s(i-1)}$.

Claim 1. *Either $\delta(i) \leq \chi(\eta, d, i)$ for every $i \in [d]$ or we can compute the claimed $Z \subseteq V(G)$ and aset of components $\mathcal{C}$ in time $\chi(\eta, d, i) * n^{\mathcal{O}}(1)$.*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Proof. Assume there is a vertex $v \in V(F)$ at depth i with more than $\chi(\eta, d, i)$ children and let C_v be that vertices' children. As a direct consequence of the definition of tree embeddings1, the pre-image $f^{-1}(v)$ of any vertex $v \in V(F)$ may only have at most d neighbours. By the pigeonhole principle, there must exist a set $U \subseteq C_v$ of size at least $\chi' = \frac{\chi}{2^{d-i} * s(i-1)}$ where for each $v_1, v_2 \in U$, (i) $|F_{v_1}| = |F_{v_2}|$ and (ii) the neighbourhood of $V(v_1)$ is the same as the neighbourhood of $V(v_2)$.

Assume there is a vertex $v \in V(F)$ at depth i with more than $\chi(\eta, d, i)$ children and let C_v be that vertices' children. The argument is based on the principle of abundance of witnesses: Consider the neighbourhoods of all induced subgraphs $\{G[F(c)] \mid c \in C_v\}$. there are at most $s(i-1)$ different possible values for $|F_{c_v}|$. Further, each $G[F(c)]$ may have a neighbourhood of size at most $d-i$ in G. Therefore there are at most 2^{d-i} possible different neighbourhoods. So due to the pigeon hole principle, there must be some set $U \subseteq C_v$, say $\{u_1, \ldots, u_{\chi'(\eta,d,i)}\}$, of children of size at least $\chi'(\eta, d, i) = \frac{\chi}{2^{(d-1)} * s(i-1)}$ that all pairwise share two properties: For each $j_1, j_2 \in [\chi'(\eta, d, i)]$, (i) $|F(c_{j_1})| = |F(c_{j_2})|$ and (ii) the neighbourhoods of $F(c_{j_1})$ and $F(c_{j_2})$ in G are the same. We pick an arbitrary $j \in [\chi'(\eta, d, i)]$ and name $V(F_{u_j})$ as Z from this point on.

We continue with a similar argumentation and consider all $F_{u_1}, \ldots, F_{u_{\chi'(\eta,d,i)}}$ and how they relate to Z. The number of possible edges inside of $G[F_{u_i}]$ is clearly bounded by $(s-i)^2$. Also, the number of possible edges from inside $G[F_{c_i}]$ to $G - G[F_{c_i}]$ is at most $(d-i) \cdot s(i-1)$. And so again by the pigeon hole principle, there must be some set of children of size at least $\chi''(\eta, d, i) = \frac{\chi'}{2^{s(i-1)^2 + (d-i) \cdot s(i-1)}}$ that are all pairwise isomorphic with each other with respect to Z. Notice that χ was chosen in a way such that $\chi'' \geq \eta$, which proves the existence of Z and η pairwise components that are pairwise isomorphic with respect to Z.

In order to compute these sets, we first find a vertex $v \in V(F)$ with more than $\chi(\eta, d, i)$ children and call C_v its child vertices. Then examine for each child $c \in C_v$ the size of F_c and the neighbourhood of $V(F_c)$ in G. From this we can find a set Z and components $\mathcal{C}'$ that are all the same size and have exactly Z as their neighbourhoods. Then we partition $\mathcal{C}'$ into maximal sets such that each set contains only components that are pairwise strongly isomorphic to Z. Since the size of each component is upper bounded by $s(i-1)$ and the number of possible witnesses is bounded by $2^{s(i-1)^2 + (d-i) * s(i-1)}$, we can brute-force the partition step. The number of witnesses is therefore bounded by $\chi(\eta, d, d)$. The running time of the whole procedure is dominated by the last step, giving a running time of $\chi(\eta, d, d) * n^{\mathcal{O}}(1)$. □

Proof of Lemma 4. Assume that $\delta(i) \leq \chi(\eta, d, i)$ for some $i \in [d]$. If we also consider that $s(i) \leq \delta(i) * s(i-1) + 1$ for every $[d]$, we define $\mu_2(\eta, d) = s(d)$. If this bound is exceeded, there must be at least one vertex having enough children to permit the process described in the proof of the above claim. Finally, by the previous claim we have that $\mu_1(\eta, d) = \chi(\eta, d, d)$. □

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

We are now ready to prove lemma 1.

Proof of Lemma 1. We can assume that G is connected. If G is not connected, then there are at most $\mathcal{M}$ connected components of G that contain a marked robot. In that case, we can execute the algorithms on each connected component individually and multiply the resulting running time with k. We start by computing an optimal tree decomposition of G. This requires $2^{\mathcal{O}(td^2)} * n^{\mathcal{O}(1)}$ time [24, 29].

The algorithm consists of two steps. The first step removes components of G, making repeated use of Corollary 1, until it reaches a size with less than $\mu_2(n_f + 3k + 1, td)$ vertices. When this happens, we have found a kernel. In the second step we brute-force all possible schedules in order to find an optimal schedule.

We define $r = n_f + 3k + 2$. Assuming G has more then $\mu_2(n_f + 3k + 1, td)$ vertices, by Lemma 4, we can find a set $Z \subseteq V(G)$ and a set $\mathcal{C} = \{C_1, \ldots, C_r\}$ of components in $G - Z$ that are pairwise strongly isomorphic with respect to Z. Then due to the fact that $r > n_f + 3k + 1$, there must be some component C_i that is fully blocked at timestep 0 and that is disjoint from $\mathcal{M}$. We use Lemma 3 and Corollary 1 to create an equivalent, smaller instance. This step is repeated until we receive an instance $\mathcal{I}' = ((G' = (V', E'), \mathcal{R}'), \ell)$ with $|V'| \leq \mu_2(n_f + 3k + 1, td)$.

An optimal schedule can then be brute-forced: In an optimal schedule, it cannot be that configurations repeat. This means that the number of all possible schedules is bounded by $\nu!$, where ν is the number of all possible configurations. Observe that ν is bounded by $|V(G')|$. In total, we can express the running time of the whole algorithm as a function $\gamma(n_f, k, td)$, as running the subroutine in Lemma 4, finding a tree embedding and brute-forcing an optimal schedule in the final step, each require a time bounded by a function of n_f, k and/or td. Finally, the energy expended in an optimal schedule is also bounded by $\gamma(n_f, k, td)$, as that is also the running time of the computation of the optimal schedule. □

4.0.2 Case 2: Many Free Vertices

We begin by noting that if $\ell \leq g(k, td)$, where g is any computable function, then the problem is FPT. This follows by an adaptation of the proof of [17, Theorem 5]:

Lemma 5. GCMP *is FPT parameterized by* $|\mathcal{M}| + \mathsf{tw}(G) + \ell$.

Proof. Toward the goal of proving the lemma, we make some guesses regarding some attributes of a solution for the given input.Then we define MSO formulas that, given the previously made guesses, can fully define the properties that a schedule must fulfill in order to be correct. The total number of guesses will be bounded by ℓ alone, allowing an algorithm to brute-force over all possible guesses. Finally, we invoke Courcelle's Theorem [2, 6] for every possible guess of attributes. Let S be the schedule we look for and let $W_1, \ldots, W_\alpha$, with $W_i = (v_1^i, \ldots, v_{d_i}^i)$ be the routes of the robots moving inside S.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

We guess (i) the number $\alpha \leq \ell$ of robots that move in S, (ii) lengths $d_1, \ldots, d_\alpha$ of the routes of the non-stationary robots, and (iii) a function $\mu : R \times R \to \{0, 1\}$, declaring whether two robots v_j^i and v_q^p from routes W_i and W_p (respectively) are identical. We further label the starting vertex of every robot red. Let $r : V \to \{0, 1\}$ be the function describing the vertices that are labelled red. We aim to express the following properties:

1. Each W_i is a walk in H.
2. For every i, j, p, q, v_j^i and v_q^p are equal if and only if $\mu(v_j^i, v_q^p) = 1$.
3. The walks W_i are pairwise non-conflicting routes.
4. Every walk W_i starts at a red vertex.
5. For each of the first $|\mathcal{M}|$ walks $W_1, \ldots, W_{|\mathcal{M}|}$, the initial and final vertices of each walk are precisely the starting and ending vertices, respectively, of some robot in $\mathcal{M}$.
6. For each red vertex appearing on any walk W_i, it must be the first vertex of some walk.

The following MSO formulas express those properties. The formula for Property 3 was split into two lines for improved readability.

$$v_j^i = v_{j+1}^i \vee adj(v_j^i, v_{j+1}^i) \qquad \forall_{i\in[\alpha], j\in[d_i-1]} \tag{4.1}$$

$$v_j^i = v_q^p \iff \mu(v_j^i, v_q^p) = 1 \qquad \forall_{i,p\in[|M|], j\in[d_i], q\in[d_p]} \tag{4.2}$$

$$v_t^i \neq v_t^p \qquad \forall_{i,p\in[\alpha], i\neq p, t\in[\min\{d_i,d_p\}]} \tag{4.3}$$

$$v_t^i = v_{t+1}^p \wedge v_{t+1}^i = v_t^p \qquad \forall_{i,p\in[\alpha], i\neq p, t\in[\min\{d_i,d_p\}-1]}$$

$$r(v_0^i) = 1 \qquad \forall_{i\in[\alpha]} \tag{4.4}$$

$$\exists_{j\in[|\mathcal{M}|]} v_0^i = s_j \wedge v_{d_i}^i = t_j \qquad \forall_{i\in[|\mathcal{M}|]} \tag{4.5}$$

$$\exists_{i\in|\mathcal{M}|} v^* = v_0^i \qquad \forall_{v^*\in V(G), r(v)=1} \tag{4.6}$$

The goal is to prove that $\mathcal{I}$ is a yes-instance if and only if there are guesses for α, numbers $d_1, \ldots, d_\alpha$, and function μ such that the resulting formula is true on H. The Forward direction: Let S be an optimal schedule for $\mathcal{I}$. From S we can automatically deduce the correct guesses for α, $d_1, \ldots, d_\alpha$ and μ. Properties 4.1 through 4.5 are clearly fulfilled, as we are dealing with a correct schedule. For Property 4.6, notice that since an arbitrary red vertex v starts out occupied, it must be vacated before a robot R_j steps on it. The property ensure that that robot is part of some walk and Property 4.3 makes sure that that walk does not conflict any other walks.

For the backward direction, assume we have guesses that make the MSO formula true and let $W_1, \ldots, W_\alpha$. We define the schedule S as the schedule corresponding to the walks

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

$W_1, \ldots, W_\alpha$. The routes do not conflict as Property 4.3 is fulfilled. Every robot in $\mathcal{M}$ starts at the correct vertex and reaches its destination as guaranteed by Property 4.4. □

Lemma 6. *Let H be a connected subgraph in a connected graph G, where G has treedepth $\mathsf{td}(G)$ and contains at least $|V(H)|$ many free vertices. There exists a polynomial-time computable schedule of length at most $2^{\mathsf{td}(G)} \cdot |V(H)|$ that frees up all the vertices in H.*

Proof. If H is fully occupied, we are done, so assume it is not. Let v be an arbitrary vertex in H that is occupied and let x be the closest (to v) free vertex in $G - H$. Vertex x must exist, as there are at least $|V(H)|$ many free free vertices in G. Let P be the shortest path from v to x. All vertices in $V(P) \cap (G - H)$, except for x, are occupied, as otherwise the assumption that x is the closest free vertex in $G - H$, breaks. Remember the process of *shifting* defined in the previous chapter. Shift the robots in P that are closest to x, towards x and continue this until you reach the first vertex in H that is unoccupied. After this process, at most $|P|$ moves have been made and another vertex in H has been freed up. By Proposition 2, the length of any shortest path between two arbitrary vertices in a graph of treedepth td is at most 2^{td}. Therefore we have that $|P| \leq 2^{td}$.

We repeat the above process until H is free of robots. Finding P, as well as shifting the robots inside of P requires polynomial time. After at most $|V(H)|$ iterations, the process finishes and it required polynomial time in total. This proves the claimed lemma. □

We use the above lemma to resolve the special case of GCMP where $|\mathcal{M}| = 1$, that is, GCMP1.

Lemma 7. *Given an instance $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M} = \{(s,t)\}, \mathcal{F}), \ell)$ of* GCMP1 *with $n_f \geq 2^{\mathsf{td}(G)}$, in polynomial time we can compute a schedule for $\mathcal{I}$ with a total travel length of at most $2^{2\mathsf{td}(G)+1}$.*

Proof. Let R_1 be the single robot in $\mathcal{M}$ and let P be a shortest path from s to t in G. Let C_P be the connected component of $G - \{s\}$ that contains P. The proof builds upon the previous result, Lemma 6. Assume the number of free vertices in C_P is less than or equal to $|V(P)|$. Otherwise we apply Lemma 6 to vacate $P - \{s\}$ in polynomial time and afterwards we move R_1 from s onto t. If the number of free vertices is smaller than $|V(P) - 1|$, we instead create a process by which we move R_1 a step back, effectively extending the length of P by one. This procedure is then repeated until P is large enough to satisfy the conditions for Lemma 6:

Remember that in any graph with treedepth k, the length of the shortest path between any two vertices has length at most 2^k. Given that $n_f \geq 2^{\{\mathsf{td}(G)\}}$ and the number of free vertices in C_P is smaller than $|V(P)| - 1$, there must be another connected component with at least one more free vertex. Let x be any free vertex in that component and let Q be the shortest path from x to v. Notice that v is a cutting vertex separating the two components. We again have that $|Q| < 2^{td(G)}$. We now shift all robots on Q, one step in

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

the direction of x. This requires at most $|Q| < 2^{td(G)}$ moves. As a result we receive an instance $\mathcal{I}'$ where s is one step further away from t. Moreover, since the v is a cutting vertex, the "new" C_P may contain further free vertices.

We repeat the above process and update P and C_P until the number of free vertices in C_P is greater than $|V(P)| - 1$. Finally we apply Lemma 6 again, making space for R_1 to pass to t. Remember that by Proposition 2, $|P| \leq 2^{td(G)}$. And so the maximum number of iterations is at most $\min\{|V(G)|, 2^{td(G)}\}$. Every iteration requires polynomial time to find a free vertex x in a new connected component, just like the final application of Lemma 6, yielding a polynomial time procedure overall.

We repeat the step-back $2^{td(G)}$ times, moving at most $2^{td(G)}$ each time. Freeing P in the end requires another $2^{td(G)} * 2^{td(G)}$ moves. The total length of the schedule is therefore at most $2^{td(G)} * 2^{td(G)} + 2^{td(G)} * 2^{td(G)} = 2^{2td(G)+1}$. □

The lemma will be helpful in the proof for our the main Lemma in this chapter. Before we prove that, we first show the following auxiliary result.

Lemma 8. *Let G be a connected graph and $p \in \mathbb{N}$ such that $|V(G)| \geq 2^{\mathsf{td}^2(G)} \cdot p^{\mathsf{td}(G)}$. Then there exists a separator S of size at most $\mathsf{td}(G)$ and a set $\mathcal{C} = \{C_1, \ldots, C_p\}$ of p many connected components of $G - S$ such that, for all $i \in [p]$, it holds that $N(C_i) = S$ and $|C_i| \leq 2^{\mathsf{td}^2(G)} \cdot p^{\mathsf{td}(G)-1}$. Moreover, we can compute S and $\mathcal{C}$ in FPT time parameterized by $p + \mathsf{td}(G)$.*

Proof. The proof works by induction on $td(G)$. In the case case, $td(G) = 2$, meaning G must be a star. Define define S as the center of the star and for components $C_1, \ldots, C_p$, pick p arbitrary leaves.

In the induction step, find a forest embedding (F, f) of G in FPT time [24, 29] and let r be the root of F. Consider the connected components of the graph $G - r$. If there are p components of size less than or equal to $2^{td(G)^2} \cdot p^{td(G)-1}$, then we set S to be r and C to be those p components, and we are done. So assume henceforth that this is not the case. Then due to the large number of vertices in G and the Pigeonhole Principle, there must be some connected component H with at least $2^{td(G)^2} \cdot p^{td(G)-1}$ vertices. Since $2^{td(G)^2} \cdot p^{td(G)-1} \geq 2^{(td(G)-1)^2} \cdot (2p)^{td(G)-2}$, we may apply the Induction Hypothesis to H. This yields a set S' in H and $2p$ components $\mathcal{C}' = \{C'_1, \ldots, C'_{2p}\}$, such that for all $i \in [2p]$ we have that $N(C_i) = S'$ and $|C'_i| \leq 2^{td(H)^2} * (2p)^{td(H)-1}$. Due to the fact that $td(H) \leq td(G) - 1$, the components each have size at most $2^{(td(G)-1)^2 * (2p)^{td(G)-2}} \leq 2^{td(G)^2} \cdot p^{td(G)-1}$. Now we partition the components into a set $\mathcal{C}'_1 = \{C'_i \mid i \in [2p], r \in N(C'_i)\}$ and $\mathcal{C}'_2 = \{C'_i \mid i \in [2p], r \notin N(C'_i)\}$. If $|\mathcal{C}_1| \leq |\mathcal{C}_2|$, we set $S = S' \cup \{r\}$ and set $\mathcal{C}$ to p arbitrary components in $\mathcal{C}_1$. Otherwise, we set $S = S'$ and again set $\mathcal{C}$ to p arbitrary components in $\mathcal{C}_2$. This finishes the proof of the lemma. □

Lemma 9. *Given an instance $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ of* GCMP *with $n_f \geq 2^{\mathsf{td}^2(G)} \cdot (3k)^{\mathsf{td}(G)}$, in* FPT*-time parameterized by $\mathsf{td} + k$ we can compute a schedule for $\mathcal{I}$ with a total travel length of at most $2^{2\mathsf{td}(G)+2} \cdot (3k + \mathsf{td}(G))$.*

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Proof. We describe an algorithm to compute a schedule as described in the lemma. The algorithm consists of these successive steps:

1. **Structure the graph**: Apply Lemma 8 to G to find a set S and $3k$ components $\mathcal{C}$ with the properties described in the lemma.
2. **Move the marked robots out of the way**: For all $i \in [k]$, move the i-th robot in $\mathcal{M}$ into $N(S) \cap C_i$, making repeated use of Lemma 7.
3. **Make space towards the destination terminals**: Create a Steiner Tree T on $S \cup \{t_i\}$ and vacate T using Lemma 6.
4. **Route the marked robots**: Move the marked robots from $N(S) \cap C_i$ through T and into $\{t_i\}$.

Step 1: Since $|V(G)| \geq 2^{td(G)^2 \cdot (3k)^{td(G)}}$, apply Lemma 8 to G in order to find a connected subgraph S with $|V(S)|$ and $3k$ components $\mathcal{C} = \{C_1, \ldots, C_{3k}\}$ of size less than or equal to $2^{td(G)^2 \cdot (3k)^{td(G)-1}}$ that all have S as their neighbourhood. Further we assume without loss of generality that $C_1, \ldots, C_{3k}$ contain no terminals, i.e. $(V(C_1) \cup \cdots \cup V(C_k)) \cap (\{s_i \mid i \in [k]\} \cup \{t_i \mid i \in [k]\}) = \emptyset$.

Step 2: Without loss of generality fix $\{R_1, \ldots, R_k\} = \mathcal{M}$. Navigate robots $R_1, \ldots, R_k$ into components $C_1, \ldots, C_k$, respectively, starting with robot R_1, then R_2, and so on. Navigating robot R_i into C_i is done by applying Lemma 7 on the subgraph $G - (C_1, \ldots, C_{i-1})$ in order to not touch the previously used components $\{C_1, \ldots, C_{i-1}\}$. Another consequence of this is that when routing robot R_i, we have that the number of free vertices in $G - (C_1, \ldots, C_{i-1})$ is still greater than or equal to $2^{td(G)^2 \cdot (3k)^{td(G)-2} * (3k-i+1) \geq 2^{td(G)}}$. This allows the repeated use of Lemma 7 as a subroutine. G remains connected over this process, as all C_i share the same neighbourhood (namely S). The total length travelled in this computation is $2^{td(G)+2}$. Finally, if a robot R_j with $j > i$ enters C_i during the process of routing R_i, we stop the subroutine immediately and simply swap the names of R_i and R_j.

Step 3: Finally we aim to move the marked robots onto all $\{t_i \mid i \in [k]\}$. For this purpose, create a Steiner Tree T of $S \cup \{t_i \mid i \in [k]\}$ in the subgraph $G - (C_1, \ldots, C_k)$[2]. This can be done in FPT time parameterized by the number of (Steiner Tree-) terminals [15]. Since every shortest path between two vertices has length at most $2^{td(G)}$ and $|S \cup \{t_i \mid i \in [k]\}| \leq td(G) + k$, we have that $|V(T)| \leq (k + td(G) - 1) \cdot 2^{td(G)}$, which is less than $n_f - \sum_{i \in [k]} |C_i| \leq 2^{td(G)^2} \cdot k \cdot (3k)^{td(G)-1}$. We can now apply Lemma 6 to T in $G - (C_1, \ldots, C_k)$ to free up all vertices in T. This requires at most $2^{td(G)} \cdot |V(T)| \leq 2^{2td(G)} \cdot (k + td(G) - 1)$ moves.

Step 4: Finally move each R_i through T and into the t_j that is free and the farthest away from S, avoiding collisions. Each robot makes at most $|V(T)|$ moves.

[2] A Steiner Tree T of a subset $U \subseteq V(G)$ of terminals is a minimum spanning tree in G that contains all terminals and potentially other vertices.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The total length of the schedule is at most $2^{2td(G)+2} \cdot k + 2^{2td(G)} \cdot (2k + td(G) - 1) + 2^{2td(G)} \cdot (2k + td(G) - 1) \leq 2^{2td(G)+2} \cdot (3k + td(G))$. □

Proof of Theorem 3. Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \ell)$ be an instance of GCMP. If $n_f \leq 2^{\mathsf{td}^2(G)} \cdot (3k)^{\mathsf{td}(G)}$ then $\mathcal{I}$ can be solved in FPT-time by Lemma 1. Otherwise, we have $n_f > 2^{\mathsf{td}^2(G)} \cdot (3k)^{\mathsf{td}(G)}$. If $\ell < 2^{2\mathsf{td}(G)+2} \cdot (3k + \mathsf{td}(G))$ then $\mathcal{I}$ can be solved in FPT-time by Lemma 5. Finally, if both n_f and ℓ are at least $2^{\mathsf{td}^2(G)} \cdot (3k)^{\mathsf{td}(G)}$, then $\mathcal{I}$ can be solved in FPT-time by Lemma 9. It follows that $\mathcal{I}$ can be solved in FPT-time, and GCMP is FPT. □

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

CHAPTER 5

Conclusions and Further Work

We investigate the parameterized complexity of a routing problem with a wide range of applications. We use a problem formalization that has recently gained attention and extend the state of the art by considering new parameterizations. The outcome is a hardness proof for one variant and a proof for inclusion in FPT for a second variant. This thesis reflects the author's contribution to a paper they co-authored with other researchers. We put special emphasis on the author's own contribution — the hardness proof. For this purpose we devise a "roadmap" featuring the engineering process of how to arrive at such a proof.

Routing problems will continue to exist in the future, and their relevance is expected to increase. Different real-world applications have different requirements, which raises the need for different problem formalizations. So for further work, one can consider additional variants of GCMP1 and GCMP. The question of how structure in the underlying graph helps in terms of complexity is an interesting research avenue. Currently, a prominent graph measure is treewidth, but the complexity of GCMP parameterized by $|\mathcal{M}|$ plus the treewidth of the input graph remains unkonwn. A starting point for this is a polynomial-time algorithm for GCMP1 on trees [27], but until now no further progress has been made in this direction. Even for $|\mathcal{M}| = 2$ no polynomial-time algorithm is known. Progress on CMP on planar graphs would be equally important.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Overview of Generative AI Tools Used

AI tools were used solely for the following purposes:

1. Grammar-/spellchecking
2. Stylistic language improvements
3. Translating the Acknowledgements, Abstract and Use of AI-tools into German

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Übersicht verwendeter Hilfsmittel

KI-Tools wurden ausschließlich für folgende Zwecke eingesetzt:

1. Grammatik-/Rechtschreibprüfung
2. Stilistische Sprachverbesserungen
3. Übersetzung der Danksagungen, der Zusammenfassung und der Verwendung von KI-Tools ins Deutsche

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Figures

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

TU WIEN Bibliothek Your knowledge hub
Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Bibliography

[1] Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *CoRR*, abs/1312.1038, 2013.

[2] Stefan Arnborg, Jens Lagergren, and Detlef Seese. Easy problems for tree-decomposable graphs. *Journal of Algorithms*, 12:308–340, 1991.

[3] Jacopo Banfi, Nicola Basilico, and Francesco Amigoni. Intractability of time-optimal multirobot path planning on 2d grid graphs with holes. *IEEE Robotics Autom. Lett.*, 2(4):1941–1947, 2017.

[4] Bahareh Banyassady, Mark de Berg, Karl Bringmann, Kevin Buchin, Henning Fernau, Dan Halperin, Irina Kostitsyna, Yoshio Okamoto, and Stijn Slot. Unlabeled multi-robot motion planning with tighter separation bounds, 2022.

[5] 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 Qiang Yang and Michael J. Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 740–746. AAAI Press, 2015.

[6] Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990.

[7] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015.

[8] Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj, Dominik Leko, and M. S. Ramanujan. Routing few robots in a crowded network. In Pat Morin and Eunjin Oh, editors, *19th International Symposium on Algorithms and Data Structures, WADS 2025, August 11-15, 2025, York University, Toronto, Canada*, volume 349 of *LIPIcs*, pages 20:1–20:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[9] Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan. Parameterized algorithms for coordinated motion planning: Minimizing energy. In

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Karl Bringmann, Martin Grohe, Gabriele Puppis, and Ola Svensson, editors, *51st International Colloquium on Automata, Languages, and Programming, ICALP 2024, July 8-12, 2024, Tallinn, Estonia*, volume 297 of *LIPIcs*, pages 53:1–53:18. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2024.

[10] Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan. Parameterized algorithms for multiagent pathfinding on trees. In Sanmay Das, Ann Nowé, and Yevgeniy Vorobeychik, editors, *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*, pages 584–592. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2025.

[11] Erik D. Demaine, Sándor P. Fekete, Phillip Keldenich, Henk Meijer, and Christian Scheffer. Coordinated motion planning: Reconfiguring a swarm of labeled robots with bounded stretch. *SIAM J. Comput.*, 48(6):1727–1762, 2019.

[12] Erik D. Demaine and Mikhail Rudoy. A simple proof that the (n2-1)-puzzle is hard. *Theor. Comput. Sci.*, 732:80–84, 2018.

[13] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[14] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

[15] Stuart E. Dreyfus and Robert A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971.

[16] Adrian Dumitrescu. Motion planning and reconfiguration for systems of multiple objects. In Sascha Kolski, editor, *Mobile Robots*, chapter 24. IntechOpen, Rijeka, 2007.

[17] Eduard Eiben, Robert Ganian, and Iyad Kanj. The parameterized complexity of coordinated motion planning. In Erin W. Chambers and Joachim Gudmundsson, editors, *39th International Symposium on Computational Geometry, SoCG 2023, June 12-15, 2023, Dallas, Texas, USA*, volume 258 of *LIPIcs*, pages 28:1–28:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[18] Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan. A minor-testing approach for coordinated motion planning with sliding robots. In Oswin Aichholzer and Haitao Wang, editors, *41st International Symposium on Computational Geometry, SoCG 2025, June 23-27, 2025, Kanazawa, Japan*, volume 332 of *LIPIcs*, pages 44:1–44:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2025.

[19] Sándor P. Fekete, Phillip Keldenich, Dominik Krupke, and Joseph S. B. Mitchell. Computing coordinated motion plans for robot swarms: The CG: SHOP challenge 2021. *ACM Journal on Experimental Algorithmics*, 27:3.1:1–3.1:12, 2022.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[20] Foivos Fioravantes, Dusan Knop, Jan Matyás Kristan, Nikolaos Melissinos, and Michal Opler. Exact algorithms and lowerbounds for multiagent path finding: Power of treelike topology. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 17380–17388. AAAI Press, 2024.

[21] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series*. Springer, Berlin, 2006.

[22] Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2):100–107, 1968.

[23] 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*, 3(4):76–88, 1984.

[24] Wojciech Nadara, Michal Pilipczuk, and Marcin Smulewicz. Computing treedepth in polynomial space and linear FPT time. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, pages 79:1–79:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022.

[25] Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

[26] Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer, 2012.

[27] Christos H. Papadimitriou, Prabhakar Raghavan, Madhu Sudan, and Hisao Tamaki. Motion planning on a graph (extended abstract). In *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, November 20-22, 1994*, pages 511–520. IEEE Computer Society, 1994.

[28] Daniel Ratner and Manfred Warmuth. The (n2-1)-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111–137, 1990.

[29] Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, and Somnath Sikdar. A faster parameterized algorithm for treedepth. In Javier Esparza, Pierre Fraigniaud, Thore Husfeldt, and Elias Koutsoupias, editors, *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 931–942. Springer, 2014.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[30] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artif. Intell.*, 219:40–66, 2015.

[31] Wikipedia contributors. 15 puzzle — Wikipedia, the free encyclopedia, 2025. [Online; accessed 04-September-2025].

[32] Jingjin Yu and Steven M. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In Marie desJardins and Michael L. Littman, editors, *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA*, pages 1443–1449. AAAI Press, 2013.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.