Routing Few Robots in a Crowded Network

Argyrios Deligkas
Department of Computer Science, Royal Holloway, University of London, Egham, UK

Eduard Eiben
Department of Computer Science, Royal Holloway, University of London, Egham, UK

Robert Ganian
Algorithms and Complexity Group, TU Wien, Austria

Iyad Kanj
School of Computing, DePaul University, Chicago, IL, USA

Dominik Leko
Algorithms and Complexity Group, TU Wien, IL, Austria

M. S. Ramanujan
Department of Computer Science, University of Warwick, UK

Abstract

In Graph Coordinated Motion Planning, we are given a graph G some of whose vertices are occupied by robots, and we are asked to route k marked robots to their destinations while avoiding collisions and without exceeding a given budget ℓ on the number of robot moves. We continue the recent investigation of the problem [ICALP 2024], focusing on the parameter k that captures the task of routing a small number of robots in a possibly crowded graph. We prove that the problem is W[1]-hard parameterized by ℓ even for $k = 1$, but fixed-parameter tractable parameterized by k plus the treedepth of G. We complement the latter algorithm with an NP-hardness reduction which shows that both parameters are necessary to achieve tractability.

2012 ACM Subject Classification Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases graph coordinated motion planning, parameterized complexity, treedepth

Digital Object Identifier 10.4230/LIPIcs.WADS.2025.20

Funding *Argyrios Deligkas*: UKRI EPSRC grant EP/X039862/1.
Robert Ganian: Projects No. 10.55776/Y1329 and 10.55776/COE12 of the Austrian Science Fund (FWF), Project No. ICT22-029 of the Vienna Science Foundation (WWTF).
Iyad Kanj: DePaul URC Grants 606601 and 350130.
M. S. Ramanujan: Engineering and Physical Sciences Research Council (EPSRC) grant EP/V044621/1.

1 Introduction

In many diverse settings, we are faced with the task of efficiently routing a set of marked robots through an environment while avoiding collisions (both between the marked robots themselves and with other robots that may be present). While the meaning of "efficient" is context-dependent, the two most-studied efficiency measures are the *makespan* (optimizing the amount of time) and the *energy* (optimizing the total amount of movement). In this article, we investigate the latter measure and consider the graph-theoretic setting studied in previous works [7, 8, 13, 14, 16, 17, 26]. More specifically, our article employs the problem definition from the recent ICALP paper on the topic [7][1]:

[1] See also the discussion of related work at the end of this section and the formal definitions in Section 2.

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, each *marked* robot $R_i \in \mathcal{M}$ is provided with an origin s_i and a destination t_i, while the *free* robots in $\mathcal{F}$ only have origins but no destinations; these typically represent movable obstacles or robots without specified destinations. At each time step, we can move one[2] robot from its current position to a neighboring unoccupied vertex, and a *schedule* is a sequence of moves that delivers all marked robots to their destinations.

While GCMP is in NP [7, 32], it remains NP-hard when restricted to the instances where $|\mathcal{M}| = 1$ (which we call GCMP1) [26], or where $|V(G)| = |\mathcal{R}| + 1$ (which generalizes the $(n^2 - 1)$-puzzle) [9, 27]. The authors of the preceding work [7] investigated the parameterized complexity of GCMP with respect to the two most natural parameterizations of the problem: the total number $|\mathcal{R}|$ of robots and the budget ℓ. In particular, they showed that:

1. GCMP1 is fixed-parameter tractable w.r.t. $|\mathcal{R}|$;
2. GCMP is fixed-parameter tractable w.r.t. $|\mathcal{R}|$ plus the treewidth of G; and
3. GCMP w.r.t. ℓ is W[1]-hard (and also in XP due to a trivial brute-force algorithm), but becomes fixed-parameter tractable when the input graphs have bounded local treewidth.

In this article, we take a refined look at the problem's parameterized complexity by considering the number $k = |\mathcal{M}|$ of marked robots as the parameter. We believe this perspective to be natural, as it better captures the setting where we need to navigate a small number of marked robots through a congested environment that may contain a large number of movable obstacles or other free robots. Given the NP-hardness of GCMP1, we cannot hope for tractability when parameterizing by k alone. Our primary goal is to understand which additional restrictions (more specifically parameterizations) allow us to solve instances of GCMP with possibly many robots, but only few marked robots.

Contributions. A natural first question in this line of enquiry is whether GCMP is fixed-parameter tractable w.r.t. $k + \ell$. We note that the reduction underlying the aforementioned W[1]-hardness result for GCMP [7, Theorem 4] requires a large number of marked robots and completely breaks if we attempt to restrict $|\mathcal{M}|$. Nevertheless, as our first result, we provide a new, multi-layered reduction which strengthens the previous lower bound:

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

Intuitively, Theorem 1 can be seen as ruling out a uniformly polynomial algorithm for routing a single robot even if the budget is bounded by a constant.

The alternative to restricting ℓ is to aim for tractability by combining k with natural graph-structural restrictions; this is the approach which yielded the bulk of the algorithmic contributions in several recent articles on the problem and its variants [7,8,13,14,16,17]. Given the prominence of the graph parameter treewidth, a first question that arises here concerns the complexity of GCMP when parameterized by k plus the treewidth of G. Unfortunately,

[2] When minimizing energy, the considered serial motion model is equivalent to the parallel one without cyclical moves.

progress here seems difficult: the polynomial-time algorithm for GCMP1 on trees is highly non-trivial [26], and whether this result can be lifted to routing two marked robots on trees is a long-standing open question in the field. In other words, even an XP-algorithm for GCMP parameterized by k in the special case of treewidth 1 would be a breakthrough.

Instead, here we propose using a stronger restriction on G –notably, its *treedepth*. Like treewidth, treedepth is a fundamental graph parameter that has close connections to the theory of sparsity [25] and has found applications as a parameter in a number of settings, ranging from space-efficient algorithms [24] through integer programming [20], model checking [19] and graph drawing [1,3]. As our main algorithmic contribution, we prove:

▶ **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.*

The proof of Theorem 2 is non-trivial and does not directly follow from any of the previously developed techniques for treedepth-based algorithms on their own. For the proof, we partition instances of interest into three groups and provide different fixed-parameter algorithms for each of the three groups:

1. **Instances where the number $|V(G)| - |\mathcal{R}|$ of free vertices is small.** Here, our main contribution is a structural result showing that every YES-instance admits a schedule whose length is bounded by a function of the combination of k, the treedepth and the number of free vertices. Towards this, we show that such an instance has an optimal schedule in which at least one robot never moves from its initial position and moreover, such an "irrelevant" robot can be computed in FPT-time. We then show how this fact can be used to design a reduction rule that ultimately produces a graph of bounded size without affecting the solution size.
2. **Instances where ℓ is small.**[3] This is by far the simplest case, as it follows by directly adapting the previous algorithm for GCMP parameterized by ℓ on graphs of bounded local treewidth [7, Theorem 5].
3. **Instances where both ℓ and the number $|V(G)| - |\mathcal{R}|$ of free vertices are large.** This case is handled by providing a constructive procedure which correctly solves every instance of GCMP. The challenge here lies in the fact that if we do not outright reject the instance, then our procedure must terminate within at most ℓ steps on all graphs of bounded treedepth, including those where the walks used by the marked robots may overlap in complicated ways.

At this point, it is natural to ask whether Theorem 2 is tight in the sense of requiring both treedepth *and* k to achieve tractability. On one hand, one can exclude fixed-parameter algorithms, as well as XP-algorithms, for GCMP w.r.t. k alone due to the known NP-hardness of GCMP1. However, none of the existing lower bounds rule out such algorithms when parameterized by treedepth alone. As our final result, we close this gap by establishing:

▶ **Theorem 3.** GCMP *is* NP*-hard even when restricted to planar graphs with treedepth at most nine.*

Theorem 3 implies that neither of the two parameters in our main algorithmic result can be dropped, and is obtained via a reduction from 3D-MATCHING that is entirely different from the one in Theorem 1 and complements other known lower bounds for coordinated motion planning on treelike graphs [16,17].

[3] To provide intuition, we use "small" and "large" to indicate whether or not a value is upper-bounded by a specific function of the parameter.

Further Related Work. Coordinated motion planning problems – sometimes also called multirobot pathfinding or robot routing problems – have been extensively studied by several research communities, including computational geometry [2,21,31] and artificial intelligence [4, 22, 29, 30]. The energy-minimization variant of the problem (i.e., the one considered in this article) also featured in the Third Computational Geometry Challenge at SoCG 2021 [15].

While the classical complexity of several variants of coordinated motion planning was settled already in the '90s [26], there has been a recent concentrated effort to obtain a deeper understanding of these problems via the lens of parameterized complexity. Apart from the previously mentioned work on GCMP [7], recent advances include a parameterized study of the setting with fast robots [14], fixed-parameter algorithms on solid grids [13], parameterized lower bounds for tree-like and other well-structured graphs [16, 17] and fixed-parameter approximation algorithms on trees [8]. We remark that the latter three articles primarily consider the task of makespan minimization in the parallel motion setting, which exhibits different complexity-theoretic behavior than the energy minimization setting targeted in our work. For instance, determining the existence of a schedule of constant makespan is NP-hard even when restricted to solid grid graphs [13], but schedules involving constant energy can be computed in polynomial time (and this can even be lifted to fixed-parameter tractability on graph classes of bounded local treewidth [7, Theorem 5]).

2 Preliminaries

All graphs considered in this paper are undirected and simple. We assume familiarity with standard graph-theoretic concepts and terminology [10] as well as with basic notions in parameterized complexity, including *fixed-parameter tractability* and W[1]-*hardness* [6,11,18]. For a subgraph H of a graph G and two vertices $u, v \in V(H)$, we denote by $\text{dist}_H(u, v)$ the length of a shortest path in H between u and v. For $n \in \mathbb{N}$, we let $[n]$ and $[n]_0$ denote the sets $\{1, \dots, n\}$ and $\{0, \dots, n\}$, respectively.

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 [5], which essentially says that problems expressible in a certain fragment of logic can be solved efficiently on graphs of bounded treewidth. Hence, we proceed by defining our other parameter of interest.

▶ **Definition 4** (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.*

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

▶ **Proposition 5** ([25]). *The treewidth of a graph G is at most its treedepth.*

▶ **Proposition 6** ([25]). *For a graph G and any vertex $v \in V(G)$, it holds that $\mathsf{td}(G) \le 1 + \max_{i\in[p]} \mathsf{td}(G_i)$, where $G_1, \dots, G_p$ are the connected components of $G - v$.*

▶ **Proposition 7** ([25]). *For a graph G, the maximum distance between any two vertices in G is at most $2^{\mathsf{td}(G)}$.*

Problem Definition. 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 $\forall r \in \{0, \ldots, t\}$, $u_r \neq v_r$. 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.

A *schedule* S for $\mathcal{R}$ is a set of pairwise non-conflicting routes $W_i, i \in [k]$, during a time interval $[0, t]$ such that at every time step $x \in [0, t-1]$, exactly one robot moves; i.e., there is $i \in [k]$ with route $W_i = (u_0, \ldots, u_t)$ such that $u_{x+1} \neq u_x$ and for every other robot $j \in [k] \setminus \{i\}$ with route $W_j = (v_0, \ldots, v_t)$, it holds $v_{x+1} = v_x$. 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 the sum of the lengths of its routes; this value is often called the *energy* in the literature (e.g., see [15]).

We denote by GCMP1 the restriction of GCMP to instances where $|\mathcal{M}| = 1$. We remark that even though GCMP is stated as as a decision problem, all the algorithms provided in this paper are constructive and can output a corresponding schedule (when it exists).

3 The Hardness Results

In this section, we establish our lower bounds. We start by excluding algorithms for GCMP parameterized by treedepth alone:

▶ **Theorem 3.** GCMP *is* NP*-hard even when restricted to planar graphs with treedepth at most nine.*

Next, we will prove that GCMP is W[1]-hard when parameterized by the energy ℓ even when we have only one marked robot, i.e., GCMP1. Hence, our results show that in order to derive any positive results it is necessary to combine the structural parameters of the graph and the number of marked robots.

To prove the W[1]-hardness for GCMP1, we provide a parameterized reduction from the W[1]-complete problem Multicolored Clique (MCC) parameterized by the number k of colors. We describe the reduction here.

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

Towards a proof for Theorem 1, we construct an instance $\mathcal{I}$ of GCMP1 given an instance $\mathcal{I}'$ of MCC. Let $\mathcal{I}'$ consist of $G' = (V', E')$ and a partitioning $(W_1 = \{w_{1,1}, \ldots, w_{1,|W_1|}\}, \ldots, W_k = \{w_{k,1}, \ldots, w_{k,|W_k|}\})$ of V' into k disjoint subsets. From this we construct an instance $\mathcal{I} = (G, \mathcal{R} = (\{(s_1, t_1)\}, \mathcal{F}), \ell)$ of GCMP1 that is a YES-instance if and only if there is a clique of size k in G'. Furthermore, ℓ is bounded by a function of k.

The main idea of the construction is to force R_1, the only robot with a destination, to gradually choose k "lanes" to traverse on his path from s_1 to t_1. Every lane corresponds to exactly one vertex in V'. The construction and the precise selection of the budget force (i) the robot to choose exactly one lane (and thus vertex) of each color and (ii) the existence of an edge between any two vertices corresponding to chosen lanes. The combination of (i) and (ii) implies a k-clique in G'. For an intuitive understanding of the reduction, see Figures 1a and 2 which illustrate the full construction through a simple example.

For the remainder of this section, we call a path *empty* (resp. *full*) if it consists solely of free (resp. *occupied*) vertices. 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 $P = (p_1, \ldots, p_n)$ 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. If an endpoint V_i is a singleton $\{v_i^*\}$, we may directly refer to v_i^* as the *endpoint* instead. We proceed with a formalization of the gadgets used in the reduction.

Decision Component. Intuitively, the *decision component* G_D is the part of the construction where R_1 will choose the aforementioned lanes, and it is depicted in reference to Figure 1b. Let a *lane* be a full path of length $k-1$. For each color W_m, the component G_D contains an induced subgraph called a *layer* consisting of $|W_m|$ many vertex-disjoint lanes – one for each vertex in W_m.

The reason each lane has length precisely $k-1$ is that each vertex on it will be used to check adjacency between the vertex it represents in G' and a chosen vertex in one of the remaining $k-1$ colors. To facilitate this intuition, we use the following indexing for the individual vertices over the layers and lanes. Refer to Figure 1b for visual aid. For $m, n \in [k]$ with $m \neq n$ and $j \in [|W_m|]$, we define $v_{m,j,n}$ as follows:

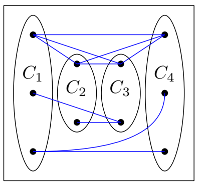
- if $m < n$, then $v_{m,j,n}$ is the $(n-1)$-st vertex in the j-th lane of the m-th layer, and
- if $m > n$, then $v_{m,j,n}$ is the n-th vertex in the j-th lane of the m-th layer.

Connect s_1 to each $\{v_{1,j,2} \mid j \in [|W_1|]\}$ with an edge. For $m \in [k-1]$, connect $\{v_{m,j,k} \mid j \in [|W_m|]\}$ and $\{v_{m+1,j,1} \mid j \in [|W_{m+1}|]\}$ with an empty path $Q_m = q_{m,1}, \ldots, q_{m,k^6}$ of length k^6. $Q = G[\{Q_m \mid m \in [k-1]\}]$ are the *tiny separators*. Let $L_m = G[\{v_{m,j,n} \mid n \in [k] \setminus \{m\}, j \in |W_m|\}]$ be the m-th *layer* and let $L = G[L_m \mid m \in [k]]$ be the graph induced on all lanes of all layers. We add two more sets of paths, the *clear paths* and the *return paths*.

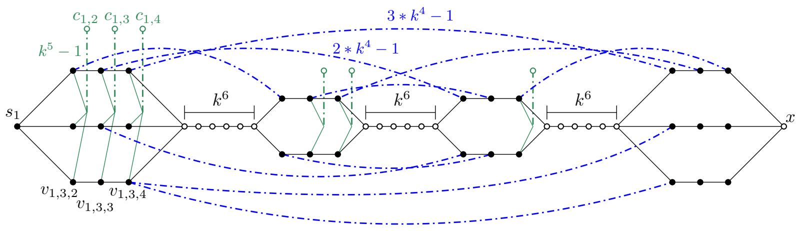
Create for every $1 \leq m < n \leq k$ a free vertex $c_{m,n}$. Then connect $\{c_{m,n}\}$ to $\{v_{m,j,n} \mid j \in |W_m|\}$ with a full path $C_{m,n}$ of length $k^5 - 1$. The set of all $C_{m,n}$ constitutes the clear paths. We define $C = \{\{c_{m,n}\} \cup C_{m,n} | 1 \leq m < n \leq k\}$.

For every edge $(w'_{m,j_1}, w'_{n,j_2}) \in E'$ with $m < n$, connect $\{v_{m,j_1,n}\}$ and $\{v_{n,j_2,m}\}$ with a full path T_{m,j_1,n,j_2} of length $k^4(n-m) - 1$. The set $T = G[\{T_{m,j_1,n,j_2} \mid (w'_{m,j_1}, w'_{n,j_2}) \in E', m < n\}]$ constitutes the return paths. Let the decision component be $G_D = G[L \cup Q \cup C \cup T]$.

Separator Component. The separator component G_S is an empty path $(x, b_2, \ldots, b_{k^{20}-1}, y)$ of length k^{20}. The first vertex x is connected to all vertices in $\{v_{k,j,k-1} \mid j \in [|W_k|]\}$. We further partition G_S into $G_{S,1} = G[\{x, b_2, \ldots, b_{k^8}\}]$ and $G_{S,2} = G[\{b_{k^8+1}, \ldots, b_{k^{20}-1}, y\}]$.



(a) The MCC instance from which to reduce.



(b) The decision component G_D plus s_1 plus the first vertex of the separator component x. The clear paths are green and the return paths are blue. Filled (resp. unfilled) vertices are occupied (resp. free) and dot dashed lines signify full paths.

Figure 1 An instance of MCC and the decision component for that instance.

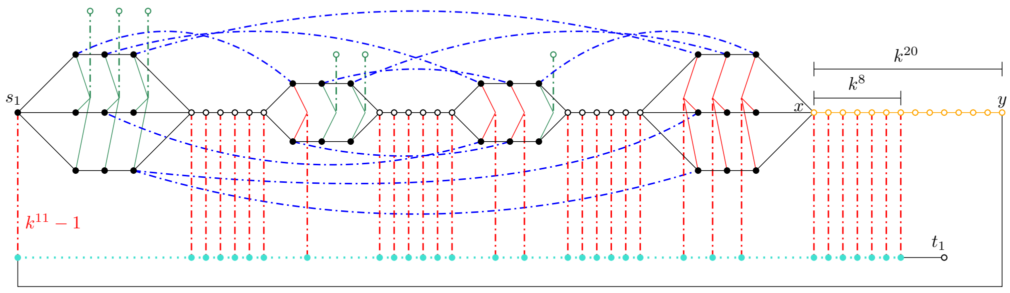


Figure 2 The full construction for the previous MCC-instance. L and Q are black, C is green, T is blue, G_S is orange and G_T is turquoise. All other edges are black. Filled (resp. unfilled) vertices are occupied (resp. free), dot dashed lines signify full paths and the dotted edges in G_T signify that those edges are each subdivided k^8 times.

Test Component. Connect $\{y\}$ and $\{t_1\}$ with a full path P of length $k^6(k-1)+\binom{k}{2}+k^8+1$. Connect vertices in P with vertices in G_D and $G_{S,1}$ with full paths consisting of $k^{11}-1$ vertices, the so-called *test paths*. To define how the test paths are connected to the remainder of the graph, partition P into the following sequence of consecutive subpaths: P_0, P_1, P'_1, P_2, P'_2, $\ldots$, P'_{k-1}, P_k, P^*_k.

1. P_0 consists of a single vertex and is connected to $\{s_1\}$ with a test path.
2. For each $m \in [k]$, we set $|P_m| = m-1$ (hence P_1 is empty and only listed for uniformity) and let $P_m = p_1, \ldots, p_{m-1}$. For $n \in [m-1]$, connect $\{v_{m,j,n} \mid j \in [|W_m|]\}$ and $\{p_n\}$ with a test path.
3. For each $m \in [k-1]$, we set $|P'_m| = k^6$ and let $P'_m = p_{m,1}, \ldots, p_{m,k^6}$. For each $i \in [k^6]$, we connect $\{p_{m_i}\}$ and $\{q_{m,i}\}$ with a test path.
4. We set $|P^*_k| = k^8$ and let $P^*_k = p_1, \ldots, p_{k^8}$. Connect $\{p_1\}$ and $\{x\}$ with a test path. Then for $i \in \{2, \ldots, k^8\}$, connect $\{p_i\}$ and $\{b_i\}$ with a test path.

Finally, subdivide every edge in P, k^8 times. The vertices created in the subdivisions should not hold robots. The resulting path is the test component G_T. Note that for $m \in [k], n \in [k] \setminus \{m\}, j \in [|W_m|]$, the vertex $v_{m,j,n} \in L$ is incident to a clear path if $m < n$. Otherwise it is incident to a test path. We set $\ell = \ell_{1,1} + \ell_{1,2} + \ell_2 + \ell_3 + \ell_4$, where the terms have the values shown in Table 1. This completes the description of the reduction.

Table 1 The respective parts of the budget used in the proof of Theorem 1.

Budget	Value	Use
$\ell_{1,1}$	$k(k-1)+k^6(k-1)+1$	Move R_1 from s_1 onto x.
$\ell_{1,2}$	$k^{20}+(k^8+1)(k^6(k-1)+\binom{k}{2}+k^8)+2$	Move R_1 from x 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.

4 Fixed-Parameter Tractability Parameterized by Treedepth + $|\mathcal{M}|$

In this section, we establish 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.*

The proof of Theorem 2 is based on a win-win argument that distinguishes whether or not the number of free vertices in the underlying graph is upper-bounded by a function of the treedepth k. We will assume henceforth that the underlying graph in the problem instance is connected; otherwise, the problem can be solved on each connected component separately.

4.1 The Case of Few Free Vertices

The main goal of this subsection is to prove the following lemma. Recall that k denotes the number of marked robots, i.e., those in $\mathcal{M}$.

▶ **Lemma 8.** *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* $\gamma(n_f,k,\mathsf{td})\cdot n^{\mathcal{O}(1)}$.

In what follows, let $\mathcal{I}=(G,\mathcal{R}=(\mathcal{M},\mathcal{F}),\ell)$ be an instance of GCMP. We assume that robots $R_1,\dots,R_k$ are the marked robots. Let S be a schedule for $\mathcal{I}$. A vertex set X is said to be *fully blocked* in S at time step t if at the end of this time step, every vertex in X is occupied by a robot from $\mathcal{F}$.

▶ **Definition 9** (Configurations). *A* pseudo-configuration *for* $\mathcal{I}$ *is a pair* (τ,Q) *where* $\tau=(\tau[1],\dots,\tau[k])$ *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.

The intuitive meaning of a configuration is that at any time step, a configuration precisely describes the positions of the robots in $\mathcal{M}$ using the tuple τ and the positions of the robots in $\mathcal{F}$ (which are essentially indistinguishable from each other) are given by the set Q. Naturally, we do not want two robots to occupy the same vertex of G, hence we require that $V(\tau)\cap Q=\emptyset$. Moreover, note that since we do not care about the destinations of the robots in $\mathcal{F}$, there could be multiple destination configurations.

▶ **Definition 10** (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* $Q_1 \Delta Q_2$[4] *has exactly two vertices* u, v *and* $uv \in E(G)$*.*

In the above definition, the first condition corresponds to moving exactly one of the robots in $\mathcal{M}$ from the vertex $\tau_1[i]$ to the vertex $\tau_2[i]$ along an edge, and the second condition corresponds to moving one of the robots in $\mathcal{F}$ between the vertices u and v while the rest remain stationary.

▶ **Definition 11** (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 12** (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})*.*

▶ **Observation 13.** *Suppose that a schedule* S *for the instance* $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \ell)$ *takes* t *time steps. For each* $s \in [t]_0$*, let* (τ_s, Q_s) *denote the configuration induced by* S *at time step* s*. Then,* $(\tau_0, Q_0), \ldots, (\tau_t, Q_t)$ *is a legal sequence of configurations. Conversely, from every legal sequence of configurations* $(\tau_0, Q_0), \ldots, (\tau_t, Q_t)$*, one can obtain a schedule* S *for* $\mathcal{I}$ *where the configuration induced by* S *at time step* s *is precisely* (τ_s, Q_s)*.*

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.

▶ **Definition 14.** *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* $v \in V(G)$*,* (i) *if* $v \in Z_1$*, then* $\phi^\star(v) = \phi(v)$*,* (ii) *if* $v \in Z_2$*, then* $\phi^\star(v) = \phi^{-1}(v)$*, and* (iii) $\phi^\star(v) = v$*, otherwise.*

The function $\phi^\star$ above simply swaps the vertices of Z_1 with the vertices of Z_2 while keeping the remaining vertices the same. In what follows (Definition 15 to Lemma 17), fix a legal sequence of configurations $\Gamma = (\tau_0, Q_0), \ldots, (\tau_t, Q_t)$ for the instance $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \ell)$.

▶ **Definition 15.** *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. For every* $i < s$*, set* $(\hat{\tau}_i, \hat{Q}_i) := (\tau_i, Q_i)$*. For every* $i \geq s$*,* (i) *set* $\hat{Q}_i := \cup_{v \in Q_i} \phi^\star(v)$*; and* (ii) *for every* $j \in [k]$*, set* $\hat{\tau}_i[j] := \phi^\star(\tau_i[j])$*.*

In words, $\hat{\Gamma}$ mimics Γ exactly, except that from time step s onward, any vertex in Z_1 (respectively, Z_2) is swapped with its image (pre-image) under ϕ.

Applying the bijection ϕ to Γ does not necessarily yield a sequence of configurations in general. However, we will demonstrate that when ϕ and the sets Z_1, Z_2 are chosen carefully, the pseudo-configurations in $\hat{\Gamma}$ are in fact, configurations – this will be useful for our analysis.

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

▶ **Definition 16.** *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 an 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 17.** *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 and both components are disjoint from the terminals of $\mathcal{M}$. 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:*
1. *The length of $\hat{\Gamma}$ is the same as that of Γ.*
2. *The sequence $\hat{\Gamma}$ is a legal sequence of configurations.*

▶ **Lemma 18.** *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:*
- *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.

The most important consequence of Lemma 18 can be easily summarized in the following, which enables us to remove an "irrelevant" vertex without affecting an optimal schedule.

▶ **Corollary 19.** *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 $\mathcal{I}' = (G - v, \mathcal{R} = (\mathcal{M} \setminus \{R\}, \mathcal{F} \setminus \{R\}), \ell)$ 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.*

We next argue that if the graph is sufficiently large, then an irrelevant vertex can be found efficiently.

▶ **Lemma 20.** *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.*

We are now ready to prove the main result of this subsection.

Proof of Lemma 8. We may assume that G is connected. Otherwise, at most k of the connected components of G can contain a robot from $\mathcal{M}$ and we simply multiply the bound obtained for a connected graph by a factor of k. Consider a tree embedding (F, f) of G of depth td. This can be computed in time $2^{\mathcal{O}(\mathsf{td}^2)} \cdot n^{\mathcal{O}(1)}$ [23, 28].

Let $r = n_f + 3k + 2$. If G has more than $\mu_2(r, \mathsf{td})$ vertices, then by Lemma 20, in time $\mu_1(r, \mathsf{td}) \cdot n^{\mathcal{O}(1)}$, we can compute a set $Z \subseteq V(G)$ and a set $\mathcal{C} = \{C_1, \ldots, C_r\}$ of components of $G - Z$ that are pairwise strongly isomorphic with respect to Z. Since r is chosen to be large enough compared to n_f and k, we may assume without loss of generality that C_r is fully blocked at time step 0 and is also disjoint from the terminals of $\mathcal{M}$.

By Lemma 18 and Corollary 19, deleting the vertices in C_r (and the robots occupying them) leads to a strictly smaller, equivalent instance. We repeat this exhaustively until we obtain an instance $\mathcal{I}' = (G', \mathcal{R}' = (\mathcal{M}', \mathcal{F}'), \ell)$ where the graph G' has at most $\mu_2(n_f + 3k + 1, \mathsf{td})$ vertices. Moreover, obtaining an optimal schedule (if one exists) for $\mathcal{I}'$ can be done by a brute-force computation on G'. Since we may assume that configurations do not repeat in an optimal schedule, the number of possible legal sequences is bounded by $\nu!$, where ν is the number of possible configurations, which is clearly bounded by a function of $|V(G')|$. ◀

4.2 The Case of Many Free Vertices

We begin by noting that if $\ell \leq g(k, \mathsf{td})$, where $g(k, \mathsf{td})$ is any computable function of k and td, then the problem is FPT. This follows by an adaptation of the proof of [7, Theorem 5]:

▶ **Lemma 21.** GCMP *is* FPT *parameterized by* $|\mathcal{M}| + \mathsf{tw}(G) + \ell$*.*

Since, it is well know that $\mathsf{td}(G) \leq tw(G)$, we get the following corollary.

▶ **Corollary 22.** GCMP *is* FPT *parameterized by* $|\mathcal{M}| + \mathsf{td}(G) + \ell$*.*

By Corollary 22 and Lemma 8, we may assume in what follows that both the budget ℓ and the number n_f of free vertices are "large". We start by establishing the following structural "subgraph-freeing" lemma:

▶ **Lemma 23.** *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.*

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

▶ **Lemma 24.** *Given an instance $\mathcal{I}$ 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 $2^{2\mathsf{td}(G)+2}$.*

Proof. Let $\mathcal{I}$ be an instance of GCMP1 whose underlying graph is G. Let R be the only robot in $\mathcal{M}$, and let s and t be its starting and destination vertices, respectively. Let P be a shortest path from s to t in G. Let C_P be the connected component of $G - s$ containing $V(P) - \{s\}$. If the number of free vertices in C_P is at least $|V(P)| - 1$, then we can apply Lemma 23 to C_P to free up all the occupied vertices in $V(P) - \{s\}$. (Note that $|V(P)| \leq 2^{\mathsf{td}(G)}$ by Proposition 7.) Afterwards, we route R from s to t. Suppose now that the number of free vertices in C_P is smaller than $|V(P)| - 1 < 2^{\mathsf{td}(G)}$. Since $n_f \geq 2^{\mathsf{td}(G)}$, there is a connected component in $G - s$ other than C_P that contains a free vertex; let v be a free vertex in $G - C_P$, and let Q be a shortest path from s to v in $G - C_P$. Note again that $|V(Q)| \leq 2^{\mathsf{td}(G)}$ by Proposition 7. Moreover, s cuts $V(Q) \setminus \{s\}$ from t. We shift all robots, including R, on $V(Q)$ by one vertex towards v. We obtain a new instance $\mathcal{I}'$ with the same underlying graph G, where the starting vertex of robot R is a neighbor s' of s. Moreover, s is a cut-vertex between s' and t, and hence this operation increases the distance between the starting vertex and the destination vertex of R. We repeat the above argument on $\mathcal{I}'$. Note that every time we are unable to free the chosen path for R, the distance between its source and destination in the current instance increases by one from the previous iteration. Since the shortest path between any two vertices in G has length at most $\min(2^{\mathsf{td}(G)}, |V(G)|)$, we repeat this operation polynomially-many times, afterwards, we can free an s-t path, and the total length traveled is at most $2^{2\mathsf{td}(G)}$. ◀

We now show how the above lemma can be employed for routing all robots with destinations, in the case where both the budget and the number of free vertices are large. Before doing that in Lemma 26, we first establish the following auxiliary result:

▶ **Lemma 25.** *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 $|N(C_i)| \leq 2^{\mathsf{td}^2(G)} \cdot p^{\mathsf{td}(G)-1}$. Moreover, we can compute S and $\mathcal{C}$ in FPT time parameterized by $p + \mathsf{td}(G)$.*

▶ **Lemma 26.** *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 $2^{2\mathsf{td}(G)+2} \cdot (3k + \mathsf{td}(G))$.*

Proof. Note that $|V(G)| \geq n_f \geq 2^{\mathsf{td}^2(G)} \cdot (3k)^{\mathsf{td}(G)}$, and hence by Lemma 25, we can find a set S of vertices of size at most $\mathsf{td}(G)$ and a family $\mathcal{C} = \{C_1, \ldots, C_{3k}\}$ of connected components in $G - S$ such that, for all $i \in [3k]$, it holds that $N(C_i) = S$ and $|C_i| \leq 2^{\mathsf{td}^2(G)} \cdot (3k)^{\mathsf{td}(G)-1}$. Without loss of generality, we can assume that, for all $i \in [k]$, C_i does not contain a starting vertex or a destination vertex of any robot in $\mathcal{M}$.

Our goal is to navigate, one-by-one, the robots in $\mathcal{M}$ to $N(S) \cap (C_1 \cup C_2 \cup \cdots \cup C_k)$ such that, when we navigate R_i to C_i, the robots $R_1, R_2, \ldots, R_{i-1}$ are already in components $C_1, C_2, \ldots, C_{i-1}$ and we perform the navigation in $G - (C_1 \cup C_2 \cup \cdots \cup C_{i-1})$; so none of the robots $R_1, \ldots, R_{i-1}$ move when we navigate robot R_i. Notice that the graph $G - (C_1 \cup C_2 \cup \cdots \cup C_{i-1})$ is connected, as each of the components $C_i, \ldots, C_{3k}$ provides the same connectivity as $(C_1 \cup C_2 \cup \cdots \cup C_{i-1})$ outside of $\mathcal{C}$, and that G is connected to start with. Hence, $G - (C_1 \cup C_2 \cup \cdots \cup C_{i-1})$ has at least $2^{\mathsf{td}^2(G)} \cdot (3k)^{\mathsf{td}(G)-1} \cdot (3k - i + 1) \geq 2^{\mathsf{td}(G)}$ many free vertices, and we can navigate R_i to C_i using Lemma 24 with total travel length at most $2^{2\mathsf{td}(G)+2}$ during this computation. However, there is one caveat. If during the execution of the schedule computed by Lemma 24, a robot R_j, for $j \in \{i+1, i+2, \ldots, k\}$, enters C_i, we stop the execution of the schedule at that point and swap the names of robots R_i and R_j. That is, robot R_j will be in C_i, and hence we manged to navigate a robot (that has not been navigated before) to C_i, and robot R_i will be navigated at a later point.

When all robots in $\mathcal{M}$ are in $N(S) \cap (C_1 \cup C_2 \cup \cdots \cup C_k)$, we compute a minimum Steiner tree T of $S \cup \{t_i \mid (s_i, t_i) \in \mathcal{M}\}$ in $G - (C_1 \cup C_2 \cup \cdots \cup C_k)$. We can compute a minimum Steiner tree in a graph in FPT-time parameterized by the number of terminals [12]. Moreover, since a shortest path between any two vertices in a graph of treedepth td has length at most 2^{td} and we are connecting $k + \mathsf{td}(G)$ many terminals, it is easy to see that $|V(T)| \leq (k + \mathsf{td}(G) - 1) \cdot 2^{\mathsf{td}(G)}$ (as we can connect one terminal to the remaining ones by shortest paths), which is less than $n_f - \sum_{i \in [k]} |C_i|$. We apply Lemma 23 to T in $G - (C_1 \cup C_2 \cup \cdots \cup C_k)$. This frees all vertices of T with a schedule of total travel length at most $2^{\mathsf{td}(G)} \cdot |V(T)| \leq 2^{2\mathsf{td}(G)} \cdot (k + \mathsf{td}(G) - 1)$. Finally, we navigate the robots in $\mathcal{M}$ inside T, one by one, to their destinations in the right order. This can done by choosing the robot whose destination in T is the farthest from S, and navigating it to its destination in T. Note that the treedepth is closed under taking induced subgraphs and $G[V(T)]$ has treedepth at most $\mathsf{td}(G)$ as well. Hence, during this final navigation each robot in $\mathcal{M}$ does at most $2^{\mathsf{td}(G)}$ many moves. Hence, the total length of the schedule is at most $2^{2\mathsf{td}(G)+2} \cdot k + 2^{2\mathsf{td}(G)} \cdot (k + \mathsf{td}(G) - 1) + k \cdot 2^{\mathsf{td}(G)} \leq 2^{2\mathsf{td}(G)+2} \cdot (3k + \mathsf{td}(G))$. ◀

Proof of Theorem 2. 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 8. 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 Corollary 22. 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 26. It follows that $\mathcal{I}$ can be solved in FPT-time, and GCMP is FPT. ◀

5 Concluding Remarks

The algorithms and lower bounds presented in this paper provide novel insights into the complexity of GCMP, specifically targeting the natural case where we need to route a small number of robots through a complicated environment. Nevertheless, we believe it is important to conclude by highlighting the prominent gaps in our understanding of the problem's complexity which remain unresolved. Even in the special case where $|\mathcal{R}| = |\mathcal{M}|$, the fixed-parameter tractability of planar GCMP when parameterized by the number $|\mathcal{R}|$ of robots remains open; here, the planar case is interesting not only because of the many usage scenarios where planar environments occur, but also because it would represent a natural generalization of the fixed-parameter tractability of the problem on solid grids [13]. In a similar vein, one might wonder whether GCMP parameterized by $|\mathcal{M}|$ is fixed-parameter or at least XP-tractable on trees – while highly non-trivial, a natural starting point in this direction is to target a polynomial-time algorithm solving the case with two marked robots on trees, which would generalize the classical algorithm for routing a single marked robot [26].

References

1 Michael J. Bannister, Sergio Cabello, and David Eppstein. Parameterized complexity of 1-planarity. *J. Graph Algorithms Appl.*, 22(1):23–49, 2018. `doi:10.7155/JGAA.00457`.

2 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. In *SoCG*, volume 224, pages 12:1–12:16, 2022. `doi:10.4230/LIPICS.SOCG.2022.12`.

3 Sujoy Bhore, Robert Ganian, Fabrizio Montecchiani, and Martin Nöllenburg. Parameterized algorithms for queue layouts. *J. Graph Algorithms Appl.*, 26(3):335–352, 2022. `doi:10.7155/JGAA.00597`.

4 Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, David Tolpin, Oded Betzalel, and Solomon Eyal Shimony. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding. In *IJCAI*, pages 740–746, 2015. URL: `http://ijcai.org/Abstract/15/110`.

5 Bruno Courcelle. The monadic second-order logic of graphs. i. recognizable sets of finite graphs. *Inf. Comput.*, 85(1):12–75, 1990. `doi:10.1016/0890-5401(90)90043-H`.

6 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer, 2015. `doi:10.1007/978-3-319-21275-3`.

7 Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj, and M. S. Ramanujan. Parameterized algorithms for coordinated motion planning: Minimizing energy. In 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. `doi:10.4230/LIPICS.ICALP.2024.53`.

8 Argyrios Deligkas, Eduard Eiben, Robert Ganian, Iyad Kanj, and Ramanujan Sridharan. Parameterized algorithms for multiagent pathfinding on trees. In *Proceedings of the 2025 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025*, 2025. to appear.

9 Erik D. Demaine and Mikhail Rudoy. A simple proof that the $(n^2 - 1)$-puzzle is hard. *Theoretical Computer Science*, 732:80–84, 2018.

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

11 Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013. doi:10.1007/978-1-4471-5559-1.

12 Stuart E. Dreyfus and Robert A. Wagner. The steiner problem in graphs. *Networks*, 1(3):195–207, 1971. doi:10.1002/NET.3230010302.

13 Eduard Eiben, Robert Ganian, and Iyad Kanj. The parameterized complexity of coordinated motion planning. In *SoCG*, volume 258, pages 28:1–28:16, 2023. doi:10.4230/LIPICS.SOCG.2023.28.

14 Eduard Eiben, Robert Ganian, Iyad Kanj, and Ramanujan Sridharan. A minor-testing approach for coordinated motion planning with sliding robots. In *SoCG*, 2025. doi:10.4230/LIPIcs.SoCG.2025.44.

15 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. doi:10.1145/3532773.

16 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*, pages 17380–17388. AAAI Press, 2024. doi:10.1609/aaai.v38i16.29686.

17 Foivos Fioravantes, Dusan Knop, Jan Matyás Kristan, Nikolaos Melissinos, and Michal Opler. Exact algorithms for multiagent path finding with communication constraints on tree-like structures. In *AAAI*, 2025. to appear. doi:10.48550/arXiv.2412.08556.

18 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*, volume XIV of *Texts in Theoretical Computer Science. An EATCS Series.* Springer, Berlin, 2006. doi:10.1007/3-540-29953-X.

19 Fedor V. Fomin, Pierre Fraigniaud, Pedro Montealegre, Ivan Rapaport, and Ioan Todinca. Distributed model checking on graphs of bounded treedepth. In Dan Alistarh, editor, *38th International Symposium on Distributed Computing, DISC 2024, October 28 to November 1, 2024, Madrid, Spain*, volume 319 of *LIPIcs*, pages 25:1–25:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2024. doi:10.4230/LIPICS.DISC.2024.25.

20 Robert Ganian and Sebastian Ordyniak. The complexity landscape of decompositional parameters for ILP. *Artif. Intell.*, 257:61–71, 2018. doi:10.1016/J.ARTINT.2017.12.006.

21 Paul Liu, Jack Spalding-Jamieson, Brandon Zhang, and Da Wei Zheng. Coordinated motion planning through randomized k-Opt (CG challenge). In *SoCG*, volume 189, pages 64:1–64:8, 2021. doi:10.4230/LIPICS.SOCG.2021.64.

22 Hang Ma, Craig Tovey, Guni Sharon, TK Kumar, and Sven Koenig. Multi-agent path finding with payload transfers and the package-exchange robot-routing problem. In *AAAI*, volume 30(1), 2016.

23 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. doi:10.4230/LIPICS.ESA.2022.79.

24 Jesper Nederlof, Michal Pilipczuk, Céline M. F. Swennenhuis, and Karol Wegrzycki. Hamiltonian cycle parameterized by treedepth in single exponential time and polynomial space. *SIAM J. Discret. Math.*, 37(3):1566–1586, 2023. doi:10.1137/22M1518943.

25 Jaroslav Nesetril and Patrice Ossona de Mendez. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics.* Springer, 2012. doi:10.1007/978-3-642-27875-4.

26 Christos H. Papadimitriou, Prabhakar Raghavan, Madhu Sudan, and Hisao Tamaki. Motion planning on a graph (extended abstract). In *STOC*, pages 511–520, 1994. doi:10.1109/SFCS.1994.365740.

27 Daniel Ratner and Manfred Warmuth. The $(n^2 - 1)$-puzzle and related relocation problems. *Journal of Symbolic Computation*, 10(2):111–137, 1990.

28 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. doi:10.1007/978-3-662-43948-7_77.

29 Pavel Surynek. An optimization variant of multi-robot path planning is intractable. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24(1), pages 1261–1263, 2010. doi:10.1609/AAAI.V24I1.7767.

30 Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015. doi:10.1016/J.ARTINT.2014.11.001.

31 Hyeyun Yang and Antoine Vigneron. A simulated annealing approach to coordinated motion planning (CG challenge). In Kevin Buchin and Éric Colin de Verdière, editors, *SoCG*, volume 189, pages 65:1–65:9, 2021. doi:10.4230/LIPICS.SOCG.2021.65.

32 Jingjin Yu and Daniela Rus. Pebble motion on graphs with rotations: Efficient feasibility tests and planning algorithms. In *WAFR*, volume 107 of *Springer Tracts in Advanced Robotics*, pages 729–746, 2014. doi:10.1007/978-3-319-16595-0_42.