Parameterized Algorithms for Coordinated Motion Planning: Minimizing Energy

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

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

Abstract

We study the parameterized complexity of a generalization of the coordinated motion planning problem on graphs, where the goal is to route a specified subset of a given set of k robots to their destinations with the aim of minimizing the total energy (i.e., the total length traveled). We develop novel techniques to push beyond previously-established results that were restricted to solid grids.

We design a fixed-parameter additive approximation algorithm for this problem parameterized by k alone. This result, which is of independent interest, allows us to prove the following two results pertaining to well-studied coordinated motion planning problems: (1) A fixed-parameter algorithm, parameterized by k, for routing a single robot to its destination while avoiding the other robots, which is related to the famous Rush-Hour Puzzle; and (2) a fixed-parameter algorithm, parameterized by k plus the treewidth of the input graph, for the standard Coordinated Motion Planning (CMP) problem in which we need to route all the k robots to their destinations. The latter of these results implies, among others, the fixed-parameter tractability of CMP parameterized by k on graphs of bounded outerplanarity, which include bounded-height subgrids.

We complement the above results with a lower bound which rules out the fixed-parameter tractability for CMP when parameterized by the total energy. This contrasts the recently-obtained tractability of the problem on solid grids under the same parameterization. As our final result, we strengthen the aforementioned fixed-parameter tractability to hold not only on solid grids but all graphs of bounded local treewidth – a class including, among others, all graphs of bounded genus.

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

Keywords and phrases coordinated motion planning, multi-agent path finding, parameterized complexity

Digital Object Identifier 10.4230/LIPIcs.ICALP.2024.53

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2404.15950`

Funding *Argyrios Deligkas*: Supported by Engineering and Physical Sciences Research Council (EPSRC) grant EP/X039862/1.
Robert Ganian: Project No. Y1329 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: Supported by Engineering and Physical Sciences Research Council (EPSRC) grants EP/V007793/1 and EP/V044621/1.

1 Introduction

The task of routing a set of robots (or agents) from their initial positions to specified destinations while avoiding collisions is of great importance in a multitude of different settings. Indeed, there is an extensive body of works dedicated to algorithms solving this task, especially in the computational geometry [1,24–26,30,32–34,37], artificial intelligence [6,35,38] and robotics [3,23,36] research communities. Such algorithms typically aim at providing a schedule for the robots which is not only safe (in the sense of avoiding collisions), but which also optimizes a certain measure of the schedule – typically its makespan or energy (i.e., the total distance traveled by all robots). In this article, we focus solely on the task of optimizing the latter of these two measures.

A common formalization of our task of interest is given by the Coordinated Motion Planning problem (also known as Multiagent Pathfinding). In the context of energy minimization, the problem can be stated as follows: Given a graph G, a budget ℓ and a set $\mathcal{M}$ of robots, each equipped with an initial vertex and destination vertex in G, compute a schedule which delivers all the robots to their destinations while ensuring that the combined length traveled of all the robots is at most ℓ. Here, a schedule can be seen as a sequence of commands to the robots, where at every time step each of the robots can move to an adjacent vertex as long as no vertex or edge is used by more than a single robot at that time step. Coordinated Motion Planning, which generalizes the famous NP-hard (n^2-1)-puzzle [11,29], has been extensively studied – both in the discrete setting considered here as well as in various continuous settings [4,6,10,14,23,28,32–36,38–40] – and has also been the target of specific computing challenges [17]. In other variants of the problem, there is no requirement to route *all* the robots to their destinations – sometimes the majority (or all but 1) of the robots are simply movable obstacles that do not have destinations of their own. This is captured, e.g., by the closely-related and well-studied Rush Hour puzzle/problem [7,19] and by Graph Motion Planning with 1 Robot (GCMP1) [27], which both feature a single robot with a designated destination.

Both Coordinated Motion Planning and GCMP1 are known to be NP-hard [11,29, 39]. While several works have already studied Coordinated Motion Planning in the context of approximation and classical complexity theory, a more fine-grained investigation of the difficulty of finding optimal schedules through the lens of *parameterized complexity* [9,13] was only carried out recently [15,18]. The work in [15] established the fixed-parameter tractability[1] of Coordinated Motion Planning parameterized by either the number k of robots or the budget ℓ (as well as the makespan variant when parameterized by k), but only on solid grids; a *solid* grid is a standard rectangular $p \times q$ grid (i.e., with no holes), for some $p, q \in \mathbb{N}$. The more recent work in [18] showed the W[1]-hardness of the makespan variant of Coordinated Motion Planning w.r.t. the number of robots. The paper [18] also showed the NP-hardness of the makespan problem-variant on trees, and presented parameterized complexity results with respect to several combinations of parameters. In this article, we focus our attention on GCMP [27], which generalizes both Coordinated Motion Planning and GCMP1 by allowing an arbitrary partitioning of the robots into those with destinations and those which act merely as movable obstacles[2].

[1] A problem is *fixed-parameter tractable* w.r.t. a parameter k if it can be solved in time $f(k) \cdot n^{\mathcal{O}(1)}$, where n is the input size and f is a computable function.

[2] While previous hardness results for GCMP considers serial motion of robots (e.g., [27]), the hardness applies to parallel motion as well. Here we obtain algorithms for the coordinated motion variant with parallel movement, but note that the results can be directly translated to the serial version.

Contribution. The aim of this article is to push our understanding of the parameterized complexity of finding energy-optimal schedules beyond the class of solid grids. While this aim was already highlighted in the aforementioned paper on solid grids [15, Section 5], the techniques used there are highly specific to that setting and it is entirely unclear how one could generalize them even to the setting of subgrids; a *subgrid* is a subgraph of a solid grid and we define its *height* to be the minimum of its two dimensions.

As our two main contributions, we provide novel fixed-parameter algorithms (1) for GCMP1 parameterized by the number of robots alone (Theorem 1), and (2) for GCMP parameterized by the number of robots plus the treewidth of the graph (Theorem 2). Theorem 2 implies, as a corollary, the fixed-parameter tractability of GCMP parameterized by k plus the minimum dimension of the subgrid.

▶ **Theorem 1.** GCMP1 *is* FPT *parameterized by the number k of robots.*

▶ **Theorem 2.** GCMP *is* FPT *parameterized by the number of robots and the treewidth of the input graph.*

The main technical tool we use to obtain both of these results is a novel fixed-parameter approximation algorithm for Generalized Coordinated Motion Planning parameterized by k alone, where the approximation error is only *additive* in k. We believe this result – summarized in Theorem 3 below – to be of independent interest.

▶ **Theorem 3.** *There is an* FPT *approximation algorithm for* GCMP *parameterized by the number k of robots which guarantees an additive error of $\mathcal{O}(k^5)$.*

The proof of Theorem 1 builds upon Theorem 3. For the proof of Theorem 2, we need to combine the approximation algorithm with novel insights concerning the "decomposability" of schedules along small separators in order to design a treewidth-based dynamic programming algorithm for the problem. A brief summary of the ideas used in this proof is provided at the beginning of Section 3.

We complement our positive results which use k as a parameter with an algorithmic lower bound showing that Coordinated Motion Planning is W[1]-hard (and hence not fixed-parameter tractable under well-established complexity-theoretic assumptions) when parameterized by the energy budget ℓ. This result (Theorem 4 below) contrasts the fixed-parameter tractability of the problem under the same parameterization when restricted to solid grids [15, Theorem 19].

▶ **Theorem 4.** GCMP *is* W[1]*-hard when parameterized by ℓ.*

While Theorem 4 establishes the intractability of the problem when parameterized by the energy on general graphs, one can in fact show that GCMP is fixed-parameter tractable under the same parameterization when the graphs are "well-structured" in the sense of having bounded *local treewidth* [16, 22]. This implies fixed-parameter tractability, e.g., on graphs of bounded genus and generalizes the aforementioned result on grids [15, Theorem 19].

▶ **Theorem 5.** GCMP *is* FPT *parameterized by ℓ on graph classes of bounded local treewidth.*

Further Related Work. As surveyed above, the computational complexity of GCMP has received significant attention, particularly by researchers in the fields of computational geometry, AI/Robotics, and theoretical computer science. The problem has been shown to remain NP-hard under a broad set of restrictions, including on graphs where only a single vertex is not occupied [21], on grids [3], and bounded-height subgrids [20]. On the other hand, a feasibility check for the existence of a schedule can be carried out in polynomial time [41]. The recent AAMAS blue sky paper [31] also highlighted the need of understanding the hardness of the problem and asked for a deeper investigation of its parameterized complexity.

2 Terminology and Problem Definition

The graphs considered in this paper are undirected simple graphs. We assume familiarity with the standard graph-theoretic concepts and terminology [12]. 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. We write $[n]$, where $n \in \mathbb{N}$, for $\{1, \ldots, n\}$.

We also assume basic familiarity with parameterized complexity theory, including *fixed-parameter tractability*, *parameterized reductions*, and the class W[1] [9,13].

Treewidth. Treewidth is a structural parameter that provides a way of expressing the resemblance of a graph to a forest. Formally, the treewidth of a graph is defined via the notion of tree decompositions as follows.

▶ **Definition 6** (Tree decomposition). A *tree decomposition* of a graph G is a pair (T, β) of a tree T and $\beta : V(T) \to 2^{V(G)}$, such that:

- $\bigcup_{t \in V(T)} \beta(t) = V(G)$,
- for any edge $e \in E(G)$, there exists a node $t \in V(T)$ such that both endpoints of e belong to $\beta(t)$, and
- for any vertex $v \in V(G)$, the subgraph of T induced by the set $T_v = \{t \in V(T) : v \in \beta(t)\}$ is a tree.

The *width* of (T, β) is $\max_{v \in V(T)}\{|\beta(v)|\} - 1$. The *treewidth* of G is the minimum width of a tree decomposition of G.

Let (T, β) be a tree decomposition of a graph G. We refer to the vertices of the tree T as *nodes*. We always assume that T is a rooted tree and hence, we have a natural parent-child and ancestor-descendant relationship among nodes in T. A *leaf* node nor a *leaf* of T is a node with degree exactly one in T which is different from the root node. All the nodes of T which are neither the root node or a leaf are called *non-leaf* nodes. The set $\beta(t)$ is called the *bag* at t. For two nodes $u, t \in T$, we say that u is a *descendant* of t, denoted $u \preceq t$, if t lies on the unique path connecting u to the root. Note that every node is its own descendant. If $u \preceq t$ and $u \neq t$, then we write $u \prec t$. For a tree decomposition (T, β) we also have a mapping $\gamma : V(T) \to 2^{V(G)}$ defined as $\gamma(t) = \bigcup_{u \preceq t} \beta(u)$.

We use the following structured tree decomposition in our algorithm.

▶ **Definition 7** (Nice tree decomposition). Let (T, β) be a tree decomposition of a graph G, where r is the root of T. The tree decomposition (T, β) is called a *nice tree decomposition* if the following conditions are satisfied.

1. $\beta(r) = \emptyset$ and $\beta(\ell) = \emptyset$ for every leaf node ℓ of T; and
2. every non-leaf node (including the root node) t of T is of one of the following types:
   - **Introduce node:** The node t has exactly one child t' in T and $\beta(t) = \beta(t') \cup \{v\}$, where $v \notin \beta(t')$.
   - **Forget node:** The node t has exactly one child t' in T and $\beta(t) = \beta(t') \setminus \{v\}$, where $v \in \beta(t')$.
   - **Join node:** The node t has exactly two children t_1 and t_2 in T and $\beta(t) = \beta(t_1) = \beta(t_2)$.

A graph class closed under vertex and edge deletion is said to have *bounded local treewidth* [16, 22] if the treewidth of each graph in the class is upper-bounded by a function of its diameter. Examples of classes of bounded local treewidth include, e.g., graphs of bounded genus [5].

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 are pairwise distinct and that all the t_i are pairwise distinct. At each time step, a robot may either move to an adjacent vertex, or stay at its current vertex, and robots may move simultaneously. 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 $x \in [0, t]$ every robot remains stationary or moves.

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 go through the same edge (in opposite directions) during 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 the sum of the lengths of its routes; this value is often called the *energy* in the literature (e.g., see [17]).

Using the introduced terminology, we formalize the GENERALIZED COORDINATED MOTION PLANNING WITH ENERGY MINIMIZATION (GCMP) problem below.

GCMP

Input:	A tuple $(G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$, where G is a graph, $\mathcal{R} = \{R_i \mid i \in [k]\}$ 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 $k, \ell \in \mathbb{N}$.
Problem:	Is there a schedule for $\mathcal{R}$ of total traveled length at most ℓ?

By observing that the feasibility check of Yu and Rus [41] transfers seamlessly to the case where some robots do not have destinations, we obtain the following.

▶ **Proposition 8** ([41]). *The existence of a schedule for an instance of* GCMP *can be decided in linear time. Moreover, if such a schedule exists, then a schedule with total length traveled of $\mathcal{O}(|V(G)|^3)$ can be computed in $\mathcal{O}(|V(G)|^3)$ time.*

Proposition 8 implies inclusion in NP, and allows us to assume henceforth that every instance of GCMP is feasible (otherwise, in linear time we can reject the instance). 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) as a witness.

3 An Additive FPT Approximation for GCMP

In this section, we give an FPT approximation algorithm for GCMP with an additive error that is a function of the number k of robots.

We start by providing a high-level, low-rigor intuition for the main result of this section. Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of GCMP. Ideally, we would like to route the robots in $\mathcal{M}$ along shortest paths to their destinations, while having the other robots move away only a "little" ($\epsilon(k)$-many steps) to unblock the shortest paths for the robots in $\mathcal{M}$. Unfortunately, this might not be possible as it is easy to observe that a free robot might have to travel a long distance in order to unblock the shortest paths of other robots. (For instance, think about the situation where a free robot is positioned in the middle of a long path of degree-2 vertices that the shortest paths traverse.) However, we will show that such a situation could only happen if the blocking robots are positioned in simple graph structures containing a long path of degree-2 vertices.

We then exploit these simple structures to "guess" in FPT-time, for each robot, a location which it visits in an optimal solution and which is in the vicinity of a safe location, called a "haven"; this haven is centered around a "nice" vertex of degree at least 3, and allows the robot to avoid any passing robot within $\epsilon(k)$ moves. We show how to navigate the robots to these havens optimally in the case of GCMP1, and with an $\epsilon(k)$ overhead in the case of GCMP. Moreover, this navigation is well-structured and can be leveraged to show that no robot in an optimal solution will visit the same vertex many times. A similar navigation takes place at the end, during the routing of the robots from their havens to their destinations.

Once we obtain such a reduced instance in which all starting positions and destinations of the robots are in havens, we can use our intended strategy to navigate each robot in $\mathcal{M}$ along a shortest path, with only an $\epsilon(k)$ overhead, which immediately gives us the approximation result and the property that no robot visits any vertex more than $\epsilon(k)$ times in an optimal solution. This latter property about the optimal solution is crucial, as we exploit it later in Section 5 to design an intricate dynamic programming algorithm over a tree decomposition of the input graph.

In addition, each free robot moves at most $\epsilon(k)$ times in the considered solution for the reduced instance. This, together with the equivalence between $\mathcal{I}$ and the reduced instance in the case of GCMP1, allow us to restrict the movement of the free robots in an optimal solution to only $\epsilon(k)$-many locations, which we use in Section 4 to obtain the FPT algorithm for GCMP1.

We start by defining the notion of a nice vertex and its haven.

▶ **Definition 9** (Nice Vertex). A vertex $w \in V(G)$ is *nice* if there exist three connected subgraphs C_1, C_2, C_3 of G such that: (i) the pairwise intersection of the vertex sets of any pair of these subgraphs is exactly w, and (ii) $|V(C_1)| \geq k+1$, $|V(C_2)| \geq k+1$, and $|V(C_3)| \geq 2$. If w is nice, let $x \in V(C_3)$ be a neighbor of w, and define the *haven* H_w of w to be the subgraph of G induced by the vertices in $\{x\} \cup V(C_1) \cup V(C_2)$ whose distance from w in H_w (i.e., $\text{dist}_{H_v}(w, u)$, for $u \in V(C_1) \cup V(C_2)$) is at most k.

For a set $S \subseteq \mathcal{R}$ of robots and a subgraph H, a *configuration* of S w.r.t. H is an injection $\iota : S \longrightarrow V(H)$. The following lemma shows that we can take the robots in a haven from any configuration to any other configuration in the haven, while incurring a total of $\mathcal{O}(k^3)$ travel (in the haven) length.

▶ **Lemma 10.** *Let w be a nice vertex, let C_1, C_2, C_3 be three subgraphs satisfying conditions (i) and (ii) of Definition 9, and let H_w be a haven for w. Then for any set of robots $S \subseteq \mathcal{R}$ in H_w with current configuration $\iota(S)$, any configuration $\iota'(S)$ with respect to H_w can be obtained from $\iota(S)$ via a sequence of $\mathcal{O}(k^3)$ moves that take place in H_w.*

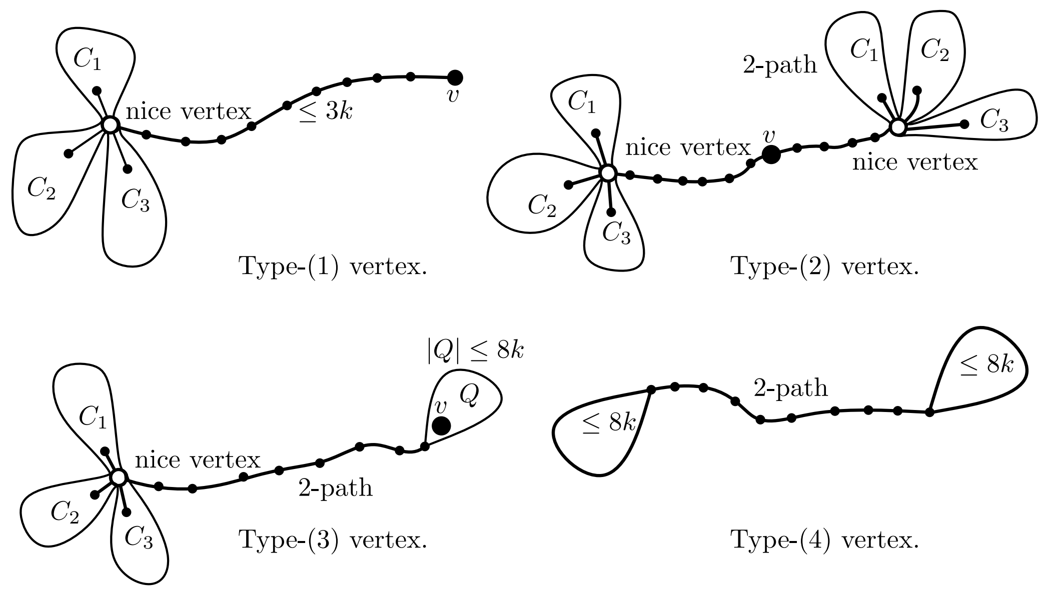

Figure 1 Illustration of the four types for a vertex v that is not nice.

Call a path P in G a *2-path* if P is an induced path of degree-2 vertices in G. The next lemma is used to characterize the possible graph structures around a vertex that is not nice; an illustration for the four cases handled by the lemma is provided in Figure 1.

▶ **Lemma 11.** *For every vertex v that is not nice, one of the following holds:*
(1) *There is a nice vertex at distance at most $3k$ from v; or*
(2) *vertex v is on a 2-path whose both endpoints are nice vertices; or*
(3) *there is a 2-path P such that one of its endpoints is nice and the other is contained in a connected subgraph Q of size at most $8k$ that P cuts from the nice vertex and such that v is in $P \cup Q$; or*
(4) *Either $|V(G)| \leq 8k$, or G consists of a 2-path P, each of whose endpoints is contained in a subgraph of size at most $8k$, such that P cuts the two subgraphs from one another.*

▶ **Definition 12** (Vertex Types). Let v be a vertex in G that is not nice. We say that v is a *type-(i) vertex*, where $i \in \{1, \ldots, 4\}$, if v satisfies Statement (i) in Lemma 11 but does not satisfy any Statement (j), where $j < i$ (if Statement (j) exists). Note that, by Lemma 11, every vertex v that is not nice must be a type-(i) vertex for some $i \in \{1, \ldots, 4\}$.

The following lemma shows that we can restrict our attention to the case where there is no type-(4) vertex in G.

▶ **Lemma 13.** *Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of* GCMP. *If there exists a type-(4) vertex $v \in V(G)$ then $\mathcal{I}$ can be solved in time $\mathcal{O}^*((4k^2 + 16k)^{2k})$ and hence is* FPT.

Before we are ready to prove the main theorem for this section, we first need the following lemma, which will allow us to restrict the movement of the free robots.

▶ **Lemma 14.** *Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of* GCMP. *For every $R_i \in \mathcal{F}$ that is within distance λ_i from a nice vertex, there is a set $B(R_i, \lambda_i)$ of vertices of cardinality $k^{\mathcal{O}(\lambda_i \cdot k + k^4)}$ such that, for any optimal solution opt($\mathcal{I}$), there exists an optimal solution opt′($\mathcal{I}$) satisfying that, in opt′($\mathcal{I}$) R_i moves only on vertices in $B(R_i, \lambda_i)$ and all the remaining robots move exactly the same in opt($\mathcal{I}$) and opt′($\mathcal{I}$). Moreover, $B(R_i, \lambda_i)$ is computable in $\mathcal{O}(|V(G)| + |E(G)|)$ time.*

▶ **Theorem 15.** *Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of* GCMP. *In* FPT*-time, we can reduce $\mathcal{I}$ to $p = g(k)$-many instances (for some computable function $g(k)$) $\mathcal{I}_1, \ldots, \mathcal{I}_p$ such that there exists $j \in [p]$ satisfying:*

1. *If $\mathcal{M} = \{R_1\}$ then $\mathcal{I}_j = (G_j, \mathcal{R}_j = (\mathcal{M}_j, \emptyset), k_j, \ell_j)$, where $k_j \leq k$ and $\ell_j \leq \ell$, is a yes-instance of* GCMP *if and only if $\mathcal{I}$ is a yes-instance. Moreover, in this case there is an optimal solution $opt(\mathcal{I}_j)$ such that $|opt(\mathcal{I}_j)| = \text{dist}_{G_j}(s_1, t_1) + \mathcal{O}(k^5)$, and for every robot $R_i \in \mathcal{M}_j \setminus \{R_1\}$, the moves of R_i in $opt(\mathcal{I}_j)$ are restricted to a vertex set of cardinality at most $k^{\mathcal{O}(k^4)}$ that is computable in linear time.*
2. *In polynomial time we can compute a schedule for $\mathcal{I}_j$ with total traveled length at most $\sum_{R_i \in \mathcal{M}_j} \text{dist}(s_i, t_i) + \mathcal{O}(k^5)$, and given such a schedule for $\mathcal{I}_j$, in polynomial time we can compute a schedule for $\mathcal{I}$ of cost at most $|opt(\mathcal{I})| + \mathcal{O}(k^5)$.*

Furthermore, every optimal schedule for $\mathcal{I}$ satisfies the property that every vertex in G is visited at most $\mathcal{O}(k^5)$ many times in the schedule.

Proof Sketch. Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of GCMP, and let $opt(\mathcal{I})$ denote an optimal schedule for $\mathcal{I}$. We give a nondeterministic reduction that produces a single instance $\mathcal{I}_j = (G_j, R_j = (\mathcal{M}_j, \emptyset), k_j, \ell_j)$, which can be made deterministic in FPT-time.

To construct $\mathcal{I}_j$, we will be making guesses about the initial and final segments of the robots' routes in $opt(\mathcal{I})$, which may lead to redefining the starting and final positions for some of them, with the purpose of getting them close to havens – as explained at the beginning of this section. The guessing is based on the types of the starting and final positions of the robots (see Lemma 11). We set $\ell_j = \ell$.

Case I. If the starting or the final position of a robot is at distance at most $11k$ from a nice vertex, we do not change it.

Case II. If the starting or the final position of a robot is a type-(2) vertex on a path P that connects two nice vertices, then we look separately at all the robots that start on P and all the robots that end on P (guessing for each free robot whether it ends on P). Let us first consider the robots that start on P. We make a guess, for each of these robots, about whether it ever leaves P, and if it does, from which endpoint of P it leaves for the first time. This splits the robots on P into three sets S_{Left}, S_{Mid}, S_{Right}. Observe that these three sets have to be "consecutive" on P (i.e., their robots respect the order of the sets), and we push the starting points of the robots in S_{Left} and in S_{Right} to distance at most k from the nice vertex they leave from for the first time. Similarly, we split the robots that end on P into the sets T_{Left}, T_{Mid}, T_{Right}. We redefine the destinations of the robots in T_{Left} and T_{Right} to vertices at distance at most k from their respective nice vertices through which they enter P, and we compute the additional travel length incurred by moving them directly from these positions at distance at most k from nice vertices to their destinations and subtract this number from ℓ_j. Recall that, according to our guess, S_{Mid} and T_{Mid} each contains precisely the robots that never leave P, and hence $S_{\text{Mid}} = T_{\text{Mid}}$. We show that for most of the robots in S_{Mid}, any optimal solution takes them directly from their starting positions to their destinations, except for those that start and end at distance at most k from the same nice vertex. For those that do not start and end at distance at most k from the same nice vertex, we compute the travel length incurred by taking them directly to their destinations, subtract it from ℓ_j, and delete these robots from $\mathcal{R}$. Finally, if $S_{\text{Mid}} \neq \emptyset$, then no robot can use P to move between its two endpoints. Moreover, if a robot is on P, then it does not need to go beyond the k-closest vertices on P to a nice vertex. Hence, if $S_{\text{Mid}} \neq \emptyset$, then we delete all the vertices on P that are at distance at least $k + 1$ from a nice vertex.

Case III. If the starting or the final position of a robot is a type-(3) vertex, which consists of a nice vertex n_R joined by a 2-path P to a vertex u in a connected subgraph C of size at most $8k$, then we start by treating this situation the same as that of a type-(2) vertex, with the role of the second nice vertex replaced by u. In particular, we guess the sets S_{Left}, S_{Mid}, S_{Right}, T_{Left}, T_{Mid}, T_{Right}, where S_{Right} (resp. T_{Right}) are the robots that start their routing by going from a vertex on P to u, or from u to a vertex on P. We also change the starting and ending positions for the robots that are not in S_{Mid} to vertices that are at distance at most k from either n_R or u; we compute the routing that reflects this change, and subtract its travel length from ℓ_j.

If $S_{\text{Mid}} \neq \emptyset$, then the vertices on which the robots in S_{Mid} are positioned separate the robots that are in $S_{\text{Right}} \cup T_{\text{Right}}$ plus the robots that start or end in C, from the rest of the graph. Note that at this point, there is no interaction between the robots in $S_{\text{Right}} \cup T_{\text{Right}} \cup S_{\text{Mid}}$ and the other robots in $\mathcal{R}$. Therefore, we can treat C and the part of P containing the robots in $S_{\text{Right}} \cup T_{\text{Right}} \cup S_{\text{Mid}}$ as a separate instance, which has the same structure as that of a type-(4) vertex. We solve this instance optimally using Lemma 13, and keep on $V(P)$ only the k-closest vertices to n_R. On the other hand, if $S_{\text{Mid}} = \emptyset$ then some robots that start (resp. end) in S_{Right} (resp. T_{Right}) or in C might also visit n_R. We show that, in the case of GCMP1, this happens only in a very restricted setting where the single robot in $\mathcal{M}$ starts or ends in C, which can be dealt with easily with the help of Lemma 13. On the other hand, if $|\mathcal{M}| \geq 2$, then this is no longer the case. We now further guess which of the robots that start (resp. end) in S_{Right} (resp. T_{Right}) or in C also visit n_R on the other end of P. Using Proposition 8, we can show that we can always route these robots at the beginning (resp. at the end) to (resp. from) a position at distance at most k from n_R with overhead at most $\mathcal{O}(k^3)$ over an optimal routing. Afterwards, we keep in G only the vertices on P at distance at most k from n_R, and remove the rest of $V(C) \cup V(P)$.

This concludes the reduction to obtain the instance $\mathcal{I}_j$ from $\mathcal{I}$.

Let us now prove Statement 1. Observe that all the free robots can now be either at distance at most $11k$ from a nice vertex and we can use Lemma 14 to compute the set $B(R_i, 11k)$ where they can move; or alternatively, they can be at distance at most k from a connected subgraph of size at most $8k$ associated with a type-(3) vertex and they will stay there, and hence, their movement in this case is restricted to $9k$ vertices.

We now show how to compute a schedule for $\mathcal{I}_j$ with total traveled length at most $\text{dist}(s_1, t_1) + \mathcal{O}(k^5)$. To do so, we first move all the robots that are at distance at most $11k$ from a nice vertex v to a haven H_v of v. We then compute a shortest path P between s_1 and t_1. We note that P can only interact with a connected subgraph C of a type-(3) vertex at the beginning or at the end. In both cases, we can use Lemma 13 to compute an optimal routing of R_1 from C (resp. to) to the first (resp. from the last) haven P intersects. Afterwards, all free robots on P are in havens. We let R_1 follow P. Whenever P interacts with a haven H that contains a free robot, we replace the subpath of P between the first interaction of R_1 with H and the last interaction of R_1 with H with a schedule computed by Lemma 10 that reconfigures H taking R_1 between its two interaction positions. Each interaction with a haven increases the length of P by $\mathcal{O}(k^3)$ and there are at most k havens that contain a free robot. Finally, we move the free robots at the end to their guessed destinations in $\mathcal{I}_j$. We can show that this incurs a travel length of $\mathcal{O}(k^5)$.

For Statement 2, if $|\mathcal{M}| = 1$ then Statement 2 follows from Statement 1. Otherwise, all starting positions and destinations are at distance at most $11k$ from nice vertices. Due to this, we can show that we can start by incurring an $\mathcal{O}(k^2)$ travel length to route each robot to a nearest haven, and finish by incurring $\mathcal{O}(k^4)$ travel length to route all the robots from havens at distance at most $11k$ from their destinations to their destinations. We can then route the robots one-by-one from a "starting" haven to a "destination" haven along a shortest path in the same manner we routed the robot R_1 in the case of GCMP1. The routing of each robot incurs an overhead of $\mathcal{O}(k^4)$ above its shortest path length, and hence the total overhead for routing all the robots between havens is $\mathcal{O}(k^5)$ and Statement 2 follows. ◀

The approximation algorithm claimed in the introduction (restated below) follows directly from Statement 2 of Theorem 15.

▶ **Theorem 3.** *There is an* FPT *approximation algorithm for* GCMP *parameterized by the number k of robots which guarantees an additive error of $\mathcal{O}(k^5)$.*

4 An FPT Algorithm for GCMP1 Parameterized by k

The aim of this section is to establish Theorem 1 by using Theorem 15.

▶ **Theorem 1.** GCMP1 *is* FPT *parameterized by the number k of robots.*

Proof Sketch. Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of GCMP1. By Statement 1 of Theorem 15, in FPT-time we can compute an equivalent instance $\mathcal{I}_j = (G_j, \mathcal{R}_j = (\mathcal{M}_j, \emptyset), k_j, \ell_j)$ to $\mathcal{I}$ satisfying that $k_j \leq k$ and $|opt(\mathcal{I}_j)| = \text{dist}_{G_j}(s_1, t_1) + \mathcal{O}(k^5)$, where $\mathcal{M} = \{R_1\}$. Moreover, for every robot $R_i \in \mathcal{M}_j \setminus \{R_1\}$, the moves of R_i in $opt(\mathcal{I}_j)$ are restricted to the vertices of a vertex-set $B(R_i)$ of cardinality at most $k^{\mathcal{O}(k^4)}$ that is computable in linear time.

Define a state graph $\mathcal{Q}$ whose vertices are k-tuples with coordinates defined as follows. The first coordinate of a k_j-tuple corresponds to R_1 and can be any of the at most n vertices in $V(G_j)$; the i-th coordinate of a tuple, where $i \in \{2, \ldots, k_j\}$, encodes the possible location of R_i and is confined to the vertices in $B(R_i)$. We purge any tuple in which a vertex in $V(G_j)$ appears in more than one coordinate of the tuple (i.e., is occupied by more than one robot in the same time step). Since $|B(R_i)| = k^{\mathcal{O}(k^4)}$, for $i \in \{2, \ldots, k_j\}$, it follows that the number of vertices in $\mathcal{Q}$ is at most $\mathcal{O}(n \cdot k^{\mathcal{O}(k^5)})$. Two vertices/states S and S' in $\mathcal{Q}$ are adjacent if there is a valid (i.e., causing no collision) single (parallel) move for the robots from their locations in S to their locations in S'; the weight of an edge (S, S') in $\mathcal{Q}$ is the number of robots in S that have moved (i.e., their positions have changed).

Define the starting configuration S_{start} in $\mathcal{Q}$ to be the k_j-tuple whose coordinates correspond to the starting positions of the robots in $\mathcal{I}_j$. Define S_{final} to be the k_j-tuple whose coordinates correspond to the destinations of the robots in $\mathcal{I}_j$. Now compute a shortest (weighted) path from S_{start} to S_{final} in $\mathcal{Q}$ (e.g., using Dijkstra's algorithm) and accept the instance $\mathcal{I}$ if and only if the weight of the computed shortest path is at most ℓ_j. The running time of the algorithm is $\mathcal{O}(|V(\mathcal{Q})|^2) = \mathcal{O}(n^2 \cdot k^{\mathcal{O}(k^5)})$, and it is clear that the above algorithm decides the instance correctly. It follows that GCMP1 is FPT parameterized by k. ◀

5 An FPT Algorithm Parameterized by Treewidth and k

In this section, we present an FPT algorithm for GCMP parameterized by the number of robots and the treewidth of the input graph combined. This result implies that the problem is FPT on certain graph classes such as graphs of bounded outerplanarity, which include subgrids of bounded height. In this sense, the result can be seen as complementary to the recently established NP-hardness of the problem on bounded-height subgrids [20].

▶ **Definition 16** (Semi-routes). A *semi-route* for R_i is a tuple $W_i = (u_0^i, \ldots, u_t^i)$ such that each u_j^i is either a vertex of G or the symbol $\perp$, and such that (i) $u_0^i = s_i$ and moreover, if $R_i \in \mathcal{M}$, then $u_t^i = t_i$, and (ii) $\forall j \in [t]$, either $u_{j-1}^i = u_j^i$ or $u_{j-1}^i u_j^i \in E(G)$ or one out of u_{j-1}^i and u_j^i is the symbol $\perp$. The notion of two conflicting semi-routes is identical to that for routes, except that we only consider time steps where neither semi-routes has $\perp$.

Intuitively, the above definition gives us a notion of partial routes, where the robots can be thought of as having become "invisible" (but still potentially in motion) for some time steps during a route (assume only a small part of the graph is visible to us). These time steps where a robot disappears from view are represented by the symbol $\perp$. Note that a robot can "reappear" at a different vertex than the one it "disappeared" at. A *semi-schedule* for $\mathcal{R}$ is a set of semi-routes during a time interval $[0, t]$ that are pairwise non-conflicting. The *length* of a semi-route and a semi-schedule is naturally defined as the number of time steps in which the robot moves from one vertex to a different vertex, and the *total traveled length* of a semi-schedule is the sum of the lengths of its semi-routes.

A *boundaried graph* [9] is a graph G with a set $X \subseteq V(G)$ of distinguished vertices called *boundary vertices*; the set X is called the *boundary* of G. A boundaried graph (G, X) is called a *p-boundaried* graph if $|X| \leq p$. A *p-boundaried subgraph* of a graph G is a p-boundaried graph (H, Z) such that (i) H is a vertex-induced subgraph of G and (ii) Z separates $V(H) \setminus Z$ from $V(G) \setminus V(H)$.

In what follows, let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of GCMP. Let (H, Z) be a p-boundaried subgraph of G with boundary Z containing all the terminals. Let S be a schedule for this instance with routes $W_i, i \in [k]$.

▶ **Definition 17** (Signatures of Schedules). Call the tuples in the set $(Z \cup \{\uparrow, \downarrow\})^k$, *configuration tuples* for (H, Z). We define the *signature* of the schedule S with respect to (H, Z) as a sequence of tuples $\tau_0, \ldots, \tau_t$, where each τ_i is a configuration tuple defined as follows: the j^{th} coordinate of τ_i signifies whether at the end of time step i

- R_j is on a vertex $v \in Z$, in which case this value is v; or
- whether it is inside H but not on the boundary, in which case this value is $\downarrow$; or
- whether it is disjoint from H, in which case this value is $\uparrow$.

The tuple τ_i is called the *signature of the schedule S with respect to (H, Z) at time step i*. The tuple τ_0 and τ_t are called *starting* and *ending* configuration tuples.

▶ **Definition 18** (Checkpoints of Schedules). We say that the schedule S:

- *externally affects* (H, Z) *at time steps* i and $i + 1$ if some robot moves from a vertex outside H to a vertex in Z during time step $i + 1$, or if some robot moves from some vertex of Z to a vertex outside H during time step $i + 1$. That is, for some robot R_j, we either have that $u_i^j \notin V(H)$ and $u_{i+1}^j \in Z$, or we have $u_i^j \in Z$ and $u_{i+1}^j \notin V(H)$.
- *internally affects* (H, Z) at time steps i and $i + 1$ if some robot moves from a vertex of H to a vertex of Z during time step $i + 1$, or if some robot moves from some vertex of Z to a vertex in H during time step $i + 1$. That is, for some robot R_j, we have $u_i^j \neq u_{i+1}^j$ and moreover, we either have that $u_i^j \in V(H)$ and $u_{i+1}^j \in Z$ or we have $u_i^j \in Z$ and $u_{i+1}^j \in V(H)$.
- *affects* (H, Z) at time steps i and $i + 1$ if it internally or externally affects (H, Z) at time steps i and $i + 1$.

The pairs of consecutive time steps at which the schedule affects (H, Z) are called *checkpoints of the schedule with respect to* (H, Z). We drop the explicit reference to (H, Z) whenever it is clear from the context. Two checkpoints $(x, x + 1)$ and $(y, y + 1)$ are said to be *consecutive* if $x < y$ and there is no checkpoint $(z, z + 1)$ such that $x < z < y$.

Roughly speaking, the checkpoints are the time steps at which the schedule interacts in some way with the separator Z. The interaction involves a robot either moving to or from a vertex of Z. As a result, the notion of checkpoints enables one to "decompose" a solution along the vertices of Z between consecutive checkpoints. We will argue that whenever Z is sufficiently small, the number of possible checkpoints will also be small.

▶ **Definition 19** (Semi-schedules). A *semi-schedule with respect to* (H, Z) is a semi-schedule where every vertex on every semi-route is contained in $V(H)$ and for every subtuple $(u, \perp, \ldots, \perp, v)$ in a semi-route, where $u, v \in V(H)$, it must be the case that $u, v \in Z$.

The intuition here is that whenever the robot "vanishes" or "reappears", it happens at the boundary Z. This allows us to analyze how a schedule interacts with the subgraph H since every movement between H and the rest of G must happen through the boundary Z.

▶ **Definition 20** (Signatures of Semi-schedules). The *signature* $\tau_0, \ldots, \tau_t$ of a semi-schedule with respect to (H, Z) is defined similarly to that of a schedule after making the assumption that $\perp$ denotes a vertex outside H, that is, we have the symbol $\uparrow$ in the j^{th} coordinate of tuple τ_i if and only if the robot R_j has $u_i^j = \perp$. A semi-schedule *externally affects* (H, Z) at time steps i and $i + 1$ if some robot "moves" from $\perp$ to Z during time step $i + 1$ or it "moves" from Z to $\perp$ during time step $i + 1$. That is, for some robot R_j, we have $u_i^j = \perp$ and $u_{i+1}^j \in Z$ or we have $u_i^j \in Z$ and $u_{i+1}^j = \perp$. The notions of internally affecting (H, Z), affecting (H, Z), checkpoints with respect to (H, Z) are defined exactly as for schedules.

▶ **Definition 21** (Checkpoint Tuples). *Checkpoint tuples* of a schedule or semi-schedule S with respect to (H, Z) are defined as those configuration tuples for (H, Z) that appear in the signature of S at the time steps participating in checkpoints of S with respect to (H, Z). The *checkpoint tuple sequence* of S with respect to (H, Z) is the set of checkpoint tuples ordered according to their respective time steps.

Our dynamic programming algorithm will make use of the following observation about signatures of schedules and semi-schedules to narrow down the search space.

▶ **Observation 22.** *Let $\tau_0, \ldots, \tau_t$ be the signature of a schedule or semi-schedule with respect to (H, Z). Then, the following properties hold:*

1. *$\tau_0 = (s_1, \ldots, s_k)$. Moreover, the coordinates of τ_t that correspond to the robots in $\mathcal{M}$ are fixed, that is, $\tau_t = (t_1, \ldots, t_{|\mathcal{M}|}, v_1, \ldots, v_{k-|\mathcal{M}|})$ for some vertices $v_1, \ldots, v_{k-|\mathcal{M}|}$ distinct from $\{t_1, \ldots, t_{|\mathcal{M}|}\}$.*
2. *The first two and last two tuples in the signature are at checkpoints. That is, $(0, 1)$ and $(t - 1, t)$ are checkpoints.*
3. *Let $(x, x + 1), (y, y + 1)$ be consecutive checkpoints such that $x + 1 < y$. Then, the tuples $\tau_{x+2}, \ldots, \tau_{y-1}$ are identical.*
4. *Let $(x, x + 1)$ be a checkpoint. There is a coordinate j such that $\tau_x[j] \neq \tau_{x+1}[j]$ and at least one of $\tau_x[j]$ or $\tau_{x+1}[j]$ is a vertex of Z.*
5. *Let $(x, x+1)$ be a checkpoint. There is no coordinate j such that $\tau_x[j] = \uparrow$ and $\tau_{x+1}[j] = \downarrow$ or $\tau_x[j] = \downarrow$ and $\tau_{x+1}[j] = \uparrow$.*
6. *For every time step x, τ_x contains at most one occurrence of any vertex of Z.*
7. *Let $(x, x + 1)$ be a checkpoint and suppose that $\tau_x[j] \neq v \in Z$ and $\tau_{x+1}[j] = v$. Then, either there is no j' such that $\tau_x[j'] = v$ or it must be the case that $\tau_{x+1}[j'] \neq v$.*
8. *Parallel movements between vertices of the boundary Z in a single time step must happen along edges that exist and moreover, no edge can be used by two robots.*

Proof. The first two statements follow from the fact that the terminals are contained in Z. The third, fourth and fifth statements follow from the definition of checkpoints and the fact that the boundary is a separator between $V(H) \setminus Z$ and $V(G) \setminus V(H)$. The sixth and seventh statements are due to the fact that no vertex can be occupied by two robots at the same time. The eighth statement is due to the fact that a schedule or semi-schedule must be comprised of pairwise non-conflicting routes or semi-routes. ◀

▶ **Observation 23.** *Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of* GCMP. *Let (H, Z) be a p-boundaried subgraph of G with boundary Z containing all the terminals. If there is a schedule S for $\mathcal{I}$ in which each vertex is entered by any robot at most $\nu(k)$ times, then the following hold.*

1. *The number of checkpoints of S with respect to (H, Z) is bounded by $\mathcal{O}(pk \cdot \nu(k))$.*
2. *The number of possible checkpoint tuple sequences of S with respect to (H, Z) is bounded by $g(k, p) = p^{\mathcal{O}(pk \cdot \nu(k))}$.*

We now proceed by defining the partial solutions which play a central role in our algorithm.

▶ **Definition 24** (Partial Solutions). Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be an instance of GCMP. Let (H, Z) be a p-boundaried subgraph of G with boundary Z containing all the terminals. Given an ordered set of configuration tuples $\gamma_1, \ldots, \gamma_q$ for (H, Z) where q is even, γ_1 is a starting configuration tuple and γ_q is an ending configuration tuple, a *partial solution corresponding to the sequence* $(\gamma_1, \gamma_2), \ldots, (\gamma_{q-1}, \gamma_q)$ is a semi-schedule S with respect to (H, Z) such that the checkpoint tuple sequence of S with respect to (H, Z) is exactly these tuples in this order. That is, the checkpoints of S with respect to (H, Z) are $(x_1, x_2), \ldots, (x_{q-1}, x_q)$ and $\tau_{x_i} = \gamma_i$ for each $i \in [q]$.

▶ **Theorem 2.** GCMP *is* FPT *parameterized by the number of robots and the treewidth of the input graph.*

Proof Sketch. Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be the given instance of GCMP. We start with a tree decomposition (T, β) of G of optimal width and add the terminals to every bag. Call the resulting tree decomposition (T, β') and let its width be w. By invoking Theorem 3 and setting $\nu(k) = \mathcal{O}(k^5)$ in Observation 23, we infer that the length of the checkpoint tuple sequence of S with respect to each boundaried graph $(G^{\downarrow}_{x,T}, \beta'(x))$ where $x \in V(T)$, is at most twice the number of checkpoints in this sequence. Hence, the length of this checkpoint tuple sequence is bounded by $\lambda(k, w) = \mathcal{O}(wk^6)$. Here, $G^{\downarrow}_{x,T}$ denotes the graph induced by the vertices that lie either in the bag $\beta'(x)$ or below it. Moreover, by the same observation, the number of possible checkpoint tuple sequences of S with respect to each boundaried graph $(G^{\downarrow}_{x,T}, \beta'(x))$ is at most $g(k, w+1) = w^{\mathcal{O}(wk^6)}$.

Based on this fact, our goal is to use dynamic programming to compute, for every $x \in V(T)$ and boundaried graph $(G^{\downarrow}_{x,T}, \beta'(x))$, and for every possible checkpoint tuple sequence of length at most $\lambda(k, w)$ with respect to this boundaried graph, the length of the best partial solution corresponding to this checkpoint tuple sequence within the boundaried graph. When x is the root node, $G^{\downarrow}_{x,T}$ is exactly the input graph G. Hence, the solution is given by the minimum entry in the table computed at the root node. Note that we only describe an algorithm to compute the length of an optimal solution. However, it is straightforward to see that an optimal schedule can also be produced in the same running time. ◀

6 Using the Total Energy as the Parameter

In the final part of our paper, we consider the complexity of GCMP in settings where we are dealing with many robots but the energy budget ℓ is small. While intuitively this may seem like an "easier" case due to the fact that ℓ immediately bounds the number of robots that will end up moving in a hypothetical schedule, we show that the problem (and even its restriction to the often-studied case where all robots have destinations) is unlikely to be fixed-parameter tractable when parameterized by ℓ.

▶ **Theorem 4.** GCMP *is* W[1]*-hard when parameterized by* ℓ.

Proof. We provide a parameterized reduction from the well-known W[1]-complete MULTICOLORED CLIQUE problem [9, 13]: decide whether a given κ-partite graph $(V_1, \dots, V_\kappa, E)$ contains a clique of size κ. To avoid any confusion, we explicitly remark that in this proof we use κ to denote the parameter of the initial instance of MULTICOLORED CLIQUE and not the number of robots in the resulting instance of GCMP.

Our reduction takes an instance $(V_1, \dots, V_\kappa, E)$ of MULTICOLORED CLIQUE and constructs an instance $(G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), \kappa', \ell)$ of GCMP as follows. To obtain G, we:

1. subdivide each edge $e \in E$ κ^3 many times;
2. attach a single new pendant vertex v' to each vertex $v \in V_1 \cup \cdots \cup V_\kappa$ in the original graph; and
3. for each pair of integers i, j such that $1 \le i < j \le \kappa$, construct a new vertex $s_{i,j}$ and make it adjacent to every vertex in V_i, and construct a new vertex $t_{i,j}$ and make it adjacent to every vertex in V_j.

For $\mathcal{R} = (\mathcal{M}, \mathcal{F})$, we set $\mathcal{F} = \emptyset$ and add two sets of robots into $\mathcal{M}$. First, for each vertex v in $V_1 \cup \cdots \cup V_\kappa$, we construct a *blocking* robot r^v and set v as its starting point as well as its destination. Second, for each of the newly constructed $s_{i,j} \in V(G)$, we construct a *clique* robot $r^{i,j}$, set $s_{i,j}$ as its starting point and $t_{i,j}$ as its destination. Finally, we set $\ell := 2\kappa + \binom{\kappa}{2} \cdot (\kappa^3 + 3)$ and $\kappa' := |\mathcal{R}|$. This completes the description of our reduction.

Towards proving correctness, we first observe that a hypothetical schedule for $(G, \mathcal{R}, \kappa', \ell)$ can never use less than $\ell := 2\kappa + \binom{\kappa}{2} \cdot (\kappa^3 + 3)$ travel length. Indeed, there are $\binom{\kappa}{2}$ many clique robots, and the shortest path between their starting and destination points has length $\kappa^3 + 3$. Moreover, every schedule must route each clique robot $r^{i,j}$ through at least one vertex in V_i and at least one vertex in V_j, and hence a hypothetical solution must spend a length of at least 2 to move at least one of the blocking robots in each V_p, $p \in [\kappa]$, out of the way and then back to its initial position (which is also its destination).

Now, assume that $(G, \mathcal{R}, \kappa', \ell)$ is a yes-instance. By the argument above, this implies that there is a schedule which moves precisely one of the blocking robots in each V_p, $p \in [\kappa]$; let us denote by w_p the starting vertex of the unique blocking robot which is moved from V_p. Since $(G, \mathcal{R}, \kappa', \ell)$ is a yes-instance, E must contain an edge between each w_i and w_j, $1 \le i < j \le \kappa$; in particular, these vertices induce a clique in $(V_1, \dots, V_\kappa, E)$.

For the converse, assume that $(V_1, \dots, V_\kappa, E)$ contains a κ-clique $w_1, \dots, w_\kappa$. We construct a solution for $(G, \mathcal{R}, \kappa', \ell)$ as follows. First, for each robot starting at w_i, $i \in [\kappa]$, we spend a length of 1 to move it to the pendant vertex w_i'. Next, we route each robot $r^{i,j}$ from $s_{i,j}$ to w_i, through the $(\kappa^3 + 1)$-length path to w_j (whereas the existence of this path is guaranteed by the existence of the edge $w_i w_j \in E$), and then to its destination $t_{i,j}$. Finally, we route each blocking robot which originally started at w_i back to w_i from w_i'. The travelled length of this schedule is precisely $2\kappa + \binom{\kappa}{2} \cdot (\kappa^3 + 3)$, as desired. ◀

As mentioned in the introduction, we complement Theorem 4 with a fixed-parameter algorithm on graph classes of bounded local treewidth.

Before proceeding to the proof, we recall the syntax of Monadic Second Order Logic (MSO) of graphs. It contains 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. It also contains 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.

The well-known Courcelle's theorem [2,8] 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. Morover, 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'.

▶ **Theorem 5.** GCMP *is* FPT *parameterized by* ℓ *on graph classes of bounded local treewidth.*

Proof. Let $\mathcal{I} = (G, \mathcal{R} = (\mathcal{M}, \mathcal{F}), k, \ell)$ be the given instance of GCMP where G belongs to a graph class that is closed under taking subgraphs and in which the treewidth of a graph of diameter d is bounded by $f(d)$ for some function f. Let $\mathcal{M}'$ denote the set of those robots in $\mathcal{M}$ whose final position is distinct from the starting position. Since each of the robots in $\mathcal{M}'$ must move at least one step, it follows that if $|\mathcal{M}'| > \ell$, then the instance is a no-instance. So, assume that $|\mathcal{M}'| \leq \ell$. Let U denote the set of starting points of the robots in $\mathcal{M}'$.

Consider an optimal schedule S. We say that an edge e appears in a route $W = (u_0, \ldots, u_t)$ in S if the endpoints of e are the vertices u_{j-1} and u_j for some $j \in [t]$. We say that e appears in S if it appears in some route in S. Let E_S denote the set of edges of G that appear in S. We observe that every connected component of the subgraph of G induced by E_S (call it G_S) contains a vertex of U. If this were not the case and there is a connected component of G_S disjoint from U, then we can delete all the routes whose edges appear in this connected component and still have a feasible schedule for the given instance, contradicting the optimality of S. Moreover, since S has total energy at most ℓ, it follows that at most ℓ edges can appear in S. Hence, it follows that S is contained in the subgraph H defined as the union of the subgraphs induced by the ℓ-balls centered at the starting positions of the robots in $\mathcal{M}'$.

Notice that since H is obtained by taking the union of at most ℓ graphs, each of diameter at most 2ℓ, it follows that H has diameter at most $2\ell^2$ and so, the treewidth of H is upper bounded by $f(2\ell^2)$. It remains to argue that GCMP is FPT parameterized by the energy and treewidth of the input graph. We do this by making some guesses, obtaining a bounded number of MSO formulas such that input is a yes-instance if and only if at least one of these formulas is true on the graph H with a label function (with constant number of labels) and then invoking Courcelle's theorem [2,8]. Let $\mathcal{M}''$ denote the subset of $\mathcal{M} \setminus \mathcal{M}'$ contained in H and let $\mathcal{R}' = \mathcal{R} \cap V(H)$.

Suppose that $\alpha \leq \ell$ robots move in S and let the routes in S be $\{W_i = (v^i_1, \ldots, v^i_{d_i}) \mid i \in [\alpha]\}$. Assume that the first $|\mathcal{M}'|$ routes, $W_1, \ldots, W_{|\mathcal{M}'|}$, correspond to the robots in $\mathcal{M}'$. We guess α and $d_1, \ldots, d_\alpha$ as these are all at most ℓ. Moreover, we guess, for every pair of vertices v^i_j and v^p_q whether they are equal. Let this be expressed by a function $\mu(v^i_j, v^p_q)$ that evaluates to 1 if and only if these vertices are guessed to be equal. Thus, even though we do not know the actual vertices in the solution (except for the starting and endpoint positions of the robots in $\mathcal{M}'$), this guess gives us the "structure" of the entire solution. Notice that the number of guesses is bounded by a function of ℓ. Now, we consider the labeled graph H where the starting points of all robots in $\mathcal{R}' \setminus \mathcal{M}''$ are labeled red and the starting points of the robots in $\mathcal{M}''$ are labeled blue. Fix a guess of α, the numbers $d_1, \ldots, d_\alpha$ and a guess of the function μ and express the following as an MSO formula.

There are ordered vertex sets $\{W_i = (v_1^i, \ldots, v_{d_i}^i) \mid i \in [\alpha]\}$ such that the following hold:

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 $\mu(v_j^i, v_q^p)$ is 1.
3. The walks W_i are pairwise non-conflicting routes.
4. For each of the first $|\mathcal{M}'|$ walks $W_1, \ldots, W_{|\mathcal{M}'|}$, the initial and final vertices are precisely the starting and ending points, respectively, of some robot in $\mathcal{M}'$.
5. For each labeled vertex appearing on any walk W_i, this vertex must be the first vertex of some walk W_i.
6. For each walk W_i starting at a blue vertex, it must terminate at the same blue vertex.

Expressing the above in MSO is straightforward to do, so we do not go in to lower level details of the MSO formula.

We next argue that the input is a yes-instance if and only if there is a guess of α, the numbers $d_1, \ldots, d_\alpha$ and a guess of the function μ such that the resulting MSO formula is true on H. Consider the forward direction. An optimal schedule S with routes $W_1, \ldots, W_\alpha$ naturally gives us a "correct" guess of α, $d_1, \ldots, d_\alpha$ and the function μ. Moreover notice that Properties 1-4 are clearly satisfied. For Property 5, notice that only the starting points of robots that never move can be disjoint from the set of initial vertices of the routes $\{W_i \mid i \in [\alpha]\}$ in the optimal schedule S. Since these robots do not move, they also cannot appear on any route W_i or they would obstruct a robot that moves. Finally, Property 6 is satisfied since every robot in $\mathcal{M}''$ that moves also ends up returning to its starting point. The converse is trivial. ◀

7 Conclusion

In this paper, we presented parameterized algorithms for fundamental coordinated motion planning problems on graphs. In particular, we proved that GCMP1 is fixed-parameter tractable parameterized by the number of robots, and that GCMP is fixed-parameter tractable parameterized by the number k of robots and the treewidth of the input graph combined. This latter result implies that GCMP is fixed-parameter tractable in several graph classes which may be of interest w.r.t. application domains, including graphs of bounded outerplanarity.

We conclude by highlighting three directions that may be pursued in future works:

1. Is GCMP FPT parameterized by k (alone) on planar graphs, or even on general graphs?
2. What is the complexity of GCMP on trees? While GCMP1 is known to be in P on trees, the complexity of the general problem on trees remains unresolved.
3. The focus of this paper was on minimizing the travel length/energy, which is one of the most-studied optimization objectives. Nevertheless, there are other important objectives, most notably that of minimizing the makespan. It seems that at least in the settings considered here, optimizing the makespan may be a computationally more challenging problem. In particular, the question of whether Coordinated Motion Planning on trees is FPT parameterized by k remains open.

References

1 Aviv Adler, Mark de Berg, Dan Halperin, and Kiril Solovey. Efficient multi-robot motion planning for unlabeled discs in simple polygons. *IEEE Transactions on Automation Science and Engineering*, 12(4):1309–1317, 2015.

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 and Automation Letters*, 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. In *SoCG*, volume 224, pages 12:1–12:16, 2022.

5 Édouard Bonnet, Jan Dreier, Jakub Gajarský, Stephan Kreutzer, Nikolas Mählmann, Pierre Simon, and Szymon Torunczyk. Model checking on interpretations of classes of bounded local cliquewidth. In Christel Baier and Dana Fisman, editors, *LICS*, pages 54:1–54:13, 2022.

6 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.

7 Josh Brunner, Lily Chung, Erik D. Demaine, Dylan H. Hendrickson, Adam Hesterberg, Adam Suhl, and Avi Zeff. 1×1 rush hour with fixed blocks is PSPACE-complete. In *FUN*, volume 157, pages 7:1–7:14, 2021.

8 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`.

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

10 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 Journal on Computing*, 48(6):1727–1762, 2019.

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

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

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

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

15 Eduard Eiben, Robert Ganian, and Iyad Kanj. The parameterized complexity of coordinated motion planning. In *SoCG*, volume 258, pages 28:1–28:16, 2023.

16 David Eppstein. Diameter and treewidth in minor-closed graph families. *Algorithmica*, 27(3):275–291, 2000.

17 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.

18 Foivos Fioravantes, Dusan Knop, Jan Matyás Kristan, Nikolaos Melissinos, and Michal Opler. Exact algorithms and lowerbounds for multiagent pathfinding: Power of treelike topology. *CoRR*, abs/2312.09646, 2023.

19 Gary William Flake and Eric B. Baum. Rush hour is PSPACE-complete, or "why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1-2):895–911, 2002.

20 Tzvika Geft and Dan Halperin. Refined hardness of distance-optimal multi-agent path finding. In *AAMAS*, pages 481–488, 2022.

21 Oded Goldreich. *Finding the shortest move-sequence in the graph-generalized 15-puzzle is NP-hard*. Springer, 2011.

22 Martin Grohe. Local tree-width, excluded minors, and approximation algorithms. *Combinatorica*, 23(4):613–632, 2003.

23 Peter E. Hart, Nils J. Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

24 John E. Hopcroft, Wolfgang J. Paul, and Leslie G. Valiant. On time versus space and related problems. In *FOCS*, pages 57–64. IEEE Computer Society, 1975.

25 John E. Hopcroft, Jacob T. Schwartz, and Micha 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.

26 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.

27 Christos H. Papadimitriou, Prabhakar Raghavan, Madhu Sudan, and Hisao Tamaki. Motion planning on a graph (extended abstract). In *STOC*, pages 511–520, 1994.

28 Geetha Ramanathan and Vangalur S. Alagar. Algorithmic motion planning in robotics: Coordinated motion of several disks amidst polygonal obstacles. In *ICRA*, volume 2, pages 514–522, 1985.

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

30 John H Reif. Complexity of the mover's problem and generalizations. In *FOCS*, pages 421–427, 1979.

31 Oren Salzman and Roni Stern. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *AAMAS*, pages 1711–1715, 2020.

32 Jacob T. Schwartz and Micha Sharir. On the piano movers' problem: III. coordinating the motion of several independent bodies: The special case of circular bodies moving amidst polygonal barriers. *The International Journal of Robotics Research*, 2:46–75, 1983.

33 Jacob T. Schwartz and Micha Sharir. On the "piano movers'" problem I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics*, 36(3):345–398, 1983.

34 Jacob T. Schwartz and Micha Sharir. On the "piano movers" problem. II. General techniques for computing topological properties of real algebraic manifolds. *Advances in Applied Mathematics*, 4(3):298–351, 1983.

35 Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

36 Irving Solis, James Motes, Read Sandström, and Nancy M. Amato. Representation-optimal multi-robot motion planning using conflict-based search. *IEEE Robotics Automation Letters*, 6(3):4608–4615, 2021.

37 Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *The International Journal of Robotics Research*, 35(14):1750–1759, 2016.

38 Glenn Wagner and Howie Choset. Subdimensional expansion for multirobot path planning. *Artificial Intelligence*, 219:1–24, 2015.

39 Jingjin Yu and Steven M. LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *AAAI*, pages 1443–1449, 2013.

40 Jingjin Yu and Steven M. LaValle. Optimal multirobot path planning on graphs: Complete algorithms and effective heuristics. *IEEE Transactions on Robotics*, 32(5):1163–1177, 2016.

41 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.