Exact Algorithms for Multiagent Path Finding with Communication Constraints on Tree-Like Structures

FOIVOS FIORAVANTES, DUŠAN KNOP, JAN MATYÁŠ KŘIŠŤAN, NIKOLAOS MELISSINOS, and MICHAL OPLER, Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Czech Republic

Consider the scenario where multiple agents have to move in an optimal way through a network, each one towards their ending position while avoiding collisions. By optimal, we mean as fast as possible, which is evaluated by a measure known as the makespan of the proposed solution. This is the setting studied in the Multiagent Path Finding problem. In this work, we additionally provide the agents with a way to communicate with each other. Due to size constraints, it is reasonable to assume that the range of communication of each agent will be limited. What should be the trajectories of the agents to, additionally, maintain a backbone of communication? In this work, we study the Multiagent Path Finding with Communication Constraints problem under the parameterized complexity framework.

Our main contribution is three exact algorithms that are efficient when considering particular structures for the input network. We provide such algorithms for the case when the communication range and the number of agents (the makespan resp.) are provided in the input and the network has a tree topology, or bounded maximum degree (has a tree-like topology, i.e., bounded treewidth resp.). We complement these results by showing that it is highly unlikely to construct efficient algorithms when considering the number of agents as part of the input, even if the makespan is 3 and the communication range is 1.

JAIR Track: Multi-Agent Path Finding

JAIR Associate Editor: Roni Stern

JAIR Reference Format:
Foivos Fioravantes, Dušan Knop, Jan Matyáš Křišťan, Nikolaos Melissinos, and Michal Opler. 2026. Exact Algorithms for Multiagent Path Finding with Communication Constraints on Tree-Like Structures. *Journal of Artificial Intelligence Research* 85, Article 36 (March 2026), 19 pages. DOI: 10.1613/jair.1.19325

1 Introduction

The Multiagent Path Finding (MAPF for short) problem is a well-known challenge in the field of planning and coordination. It involves navigating multiple agents through a topological space, often modeled as an undirected graph, to reach their respective destinations. In many real-world scenarios, additional constraints on the agents' movements are required. One such constraint is the *communication* constraint, which requires agents to maintain a connected set of vertices in a communication graph as they move; this is then the Multiagent Path Finding with Communication Constraints (MAPFCC for short) problem. This requirement can arise, for example, from the need to constantly communicate with a human operator Amigoni et al. (2017). Sometimes, only a periodic connection might be sufficient Hollinger and Singh (2012). On the other hand, applications in a video

Authors' Contact Information: Foivos Fioravantes, ORCID: 0000-0001-8217-030X, foivos.fioravantes@fit.cvut.cz; Dušan Knop, ORCID: 0000-0003-2588-5709, dusan.knop@fit.cvut.cz; Jan Matyáš Křišťan, ORCID: 0000-0001-6657-0020, kristja6@fit.cvut.cz; Nikolaos Melissinos, ORCID: 0000-0002-0864-9803, nik.melissinos@gmail.com; Michal Opler, ORCID: 0000-0002-4389-5807, michal.opler@fit.cvut.cz, Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic.

DOI: 10.1613/jair.1.19325

game movement of agents Snape et al. (2012) should require near-connectivity, since we want the group of virtual soldiers to move in a mob.

It should also be noted that the communication constraints we consider are born as a natural first step towards further understanding and providing new insights into solving the MAPF problem in the distributed setting. We believe that such a setting, where each agent needs to do some local computation taking into account only a partial view of the network and the subset of the other agents that are within its communication range, is rather natural and worth investigating. Such a framework is particularly well-suited for exploring the emergence of swarm intelligence through agent cooperation.

The complexity of the Multiagent Path Finding problem increases significantly when the movement and communication graphs are independent of each other. In fact, under these conditions, the problem is PSPACE-complete Tateo et al. (2018). This raises a natural question: Is the problems' complexity equally severe when the movement and communication graphs are related? For instance, if we assume that communication among agents occurs within the same space they are navigating, it is reasonable to model the communication graph as identical to the movement graph. Alternatively, we could consider scenarios where the communication graph is a derivative of the movement graph, such as its third power, allowing agents to communicate over a distance of three edges in the original graph. However, the problem stays PSPACE-complete even if the agents move in a subgraph of a 3D grid and the communication is based on radius Calviac et al. (2023). We refer the reader to the next section for the formal definitions.

Both Multiagent Path Finding and Multiagent Path Finding with Communication Constraints problems are systematically studied; most researchers deal with the hardness using specific algorithms or heuristics. The most popular approaches to find optimal solutions are using the A* algorithm (e.g. Sharon et al. (2015, 2013)) or ILP solvers (e.g. Yu and LaValle (2013)). Another popular line of research used the Picat language Zhou et al. (2015); Barták et al. (2017). A wide range of heuristics is commonly used, such as those based on local search (WHCA*, ECBS), SAT solvers (e.g. Surynek et al. (2022)), or reinforcement learning (e.g. Gupta et al. (2017)) to name just a few. The WHCA* (Windowed Hierarchical Cooperative A*) approach was described and analyzed by Silver Silver (2005); Korf (1990). The ECBS (Enhanced Conflict-Based Search) approach was used by Barer et al. (2014); similarly for the Improved CBS Boyarski et al. (2015). For more related references, the reader might visit some of the more recent surveys on this subfield Felner et al. (2017); Surynek (2022); Stern et al. (2019).

Since our paper deals with Multiagent Path Finding with Communication Constraints from a theoretical point of view, we are mostly interested in the computational complexity study of it. The study of similar-nature problems started long ago Wilson (1974); Kornhauser et al. (1984); Goldreich (2011) and was mostly related to puzzle games. Many of these games were shown to be PSPACE-complete Hearn and Demaine (2005). Surynek (2010) provided a direct proof that Multiagent Path Finding is NP-hard. We stress here that the most "direct" argument for NP-membership (i.e., by providing a solution) does not often work for MAPFCC-alike problems since some agents might need to revisit some vertices in every optimal solution. Consequently, solutions that minimize the makespan may require superpolynomial time. Last but not least, Eiben et al. (2023); Fioravantes et al. (2024) studied the parameterized complexity of MAPF and provided initial results for tree-like topology G.

Our Contributions. We study the complexity of the MAPFCC problem from the viewpoint of parameterized algorithms Downey and Fellows (2012). As the main parameter, we select the number of agents k and prove that MAPFCC is W[1]-hard even if the makespan ℓ is 3, and the communication range d is 1 (on the input graph); see Theorem 1. In particular, this means that any single parameter among k, ℓ and/or d is highly unlikely to lead to an FPT(fixed-parameter tractable) algorithm. The same holds true for the combined parameters $k + d$, $k + \ell$, $\ell + d$ and $k + d + \ell$. We contrast this extremely negative result with algorithms that manage to escape its hardness.

We show that if we parameterize jointly by the number of agents, the maximum degree of the input graph, and the communication range, then the size of the configuration network is bounded by a function of these parameters (Theorem 2). Therefore, the problem is in FPT and so is the size of the configuration network that is often used in the A*-based approaches to MAPF. Next, we show that if the input graph G is a tree, we can obtain an FPT algorithm with respect to the number of agents plus the communication range (Theorem 3). Intuitively, the idea is to use the communication range to prune the input tree and invoke Theorem 2.

It is natural to try to leverage the algorithms from trees to graph families that have bounded treewidth. We do this for a slightly different combination of parameters. That is, MAPF is FPT for the combination of the treewidth of the graph G, makespan, and the communication range (Theorem 4). This result is achieved by bounding the treewidth of the so-called augmented graph—introduced by Ganian et al. (2021)—which adds edges between the start and end vertices of each agent. We then formulate MAPF using a Monadic Second Order (MSO) logic formula which can be decided by the result of Courcelle (1990).

This result not only highlights the structure of the augmented graph, which could be of independent interest in future research, but also suggests the potential utility of generic MSO solvers (e.g., Langer (2013); Bannach and Berndt (2019); Hecher (2023)) for practical applications. Moreover, by parameterizing with the number of agents, we extend our results to scenarios based on the local treewidth rather than the global treewidth (Corollary 2). Despite this being a rather technical parameter, it does lead to pertinent results. For example, we obtain an FPT algorithm for planar graphs with respect to the number of agents, makespan, and the communication range (Corollary 3). We stress here that many minor-closed graph classes have a bounded local treewidth; the class of planar graphs is just a single representative. Note that this is in contrast to Theorem 1 as the parameterization is the same and the only difference is the graph class.

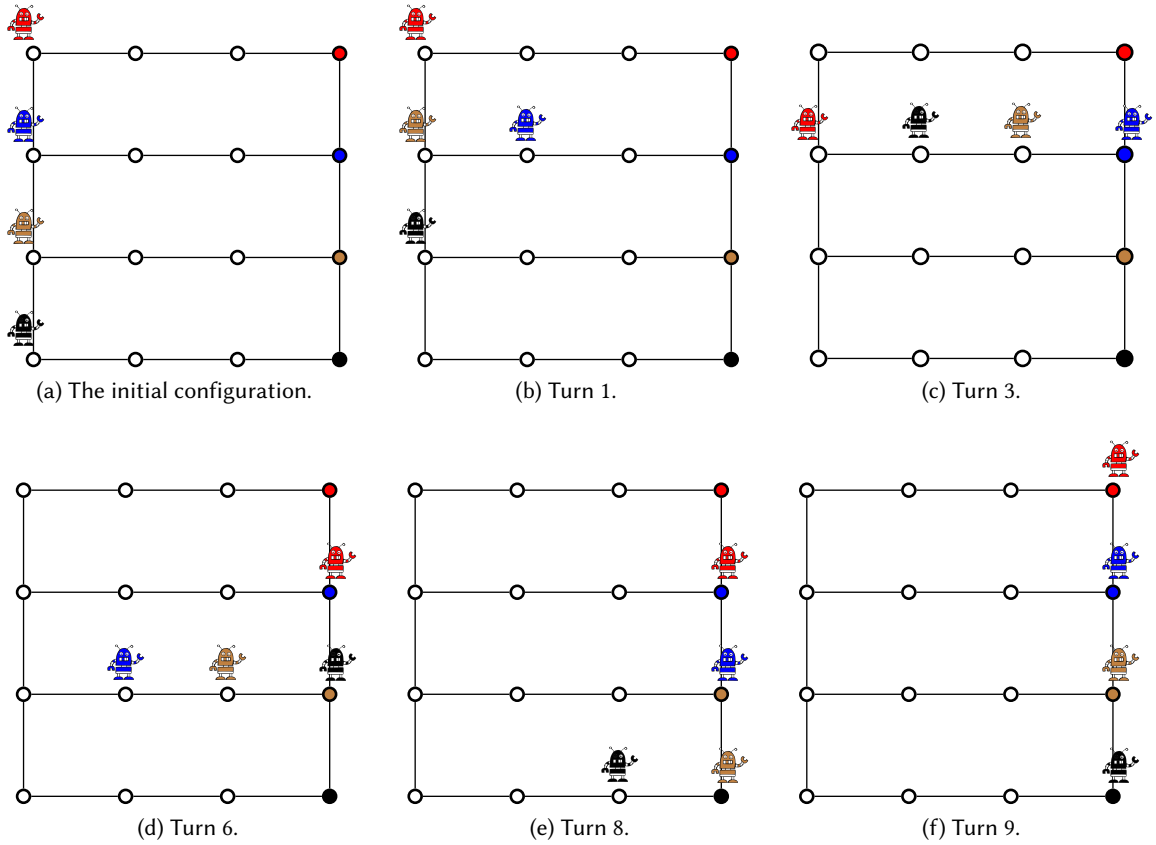
2 Preliminaries

Formally, the input of the Multiagent Path Finding problem consists of a graph $G = (V, E)$, a set of agents A, two functions $s_0 \colon A \to V$, $t \colon A \to V$ and a positive integer ℓ, known as the makespan. The functions s_0 and t serve as the starting and target vertices of the agents, respectively. For any pair $a, b \in A$ where $a \neq b$, we have that $s_0(a) \neq s_0(b)$ and $t(a) \neq t(b)$. Initially, each agent $a \in A$ is placed on the vertex $s_0(a)$. The schedule $s_0, s_1, \ldots, s_\mu$ assigns each agent a vertex in the given turn $i \in [\mu]$. In a specific turn, agents are allowed to move to a vertex neighboring their position in the previous turn, but are not obliged to do so. The agents can make at most one move per turn, and each vertex can host at most one agent at a given turn. The position of the agents at the end of the turn i (after the agents have moved) is given by an injective function $s_i \colon A \to V$. It is worth mentioning that there are two main variants of the classical Multiagent Path Finding problem, according to whether *swaps* are allowed or not. A swap between two agents a and b during a turn i is defined as the behavior where $s_{i+1}(a) = s_i(b)$ and $s_{i+1}(b) = s_i(a)$, with $s_i(a)$ and $s_i(b)$ being adjacent. In other words, a swap happens when two agents start from adjacent positions and exchange them within one turn.

In this paper, we consider the Multiagent Path Finding with Communication Constraints (MAPFCC for short) problem. In this generalization of the classical Multiagent Path Finding problem, each agent has the ability to communicate with other agents that are located within his *communication range*, and it must always be ensured that there is a subset of agents that form a backbone ensuring the communication between all pairs of agents. This communication range is modeled by an integer d that is additionally part of the input.

In order to define what is a feasible solution for the connected variant, we first need to define an auxiliary graph D; let us call this the *communication* graph. First, we set $V(D) = V(G)$. Then, for every pair $u, v \in V(D)$ we add an edge in D if and only if $\mathrm{dist}_G(u, v) \leq d$. We say that a vertex set $W \subseteq V(G)$ is *d-connected* if the induced subgraph $D[W]$ is connected. We say that a sequence $s_1, \ldots, s_\mu$ is a *feasible solution* of $\langle G, A, s_0, t, d, \ell \rangle$ if:

(1) $s_i(a)$ is a neighbor of $s_{i-1}(a)$ in G, for every agent $a \in A$, $i \in [\mu]$,



(a) The initial configuration. (b) Turn 1. (c) Turn 3.

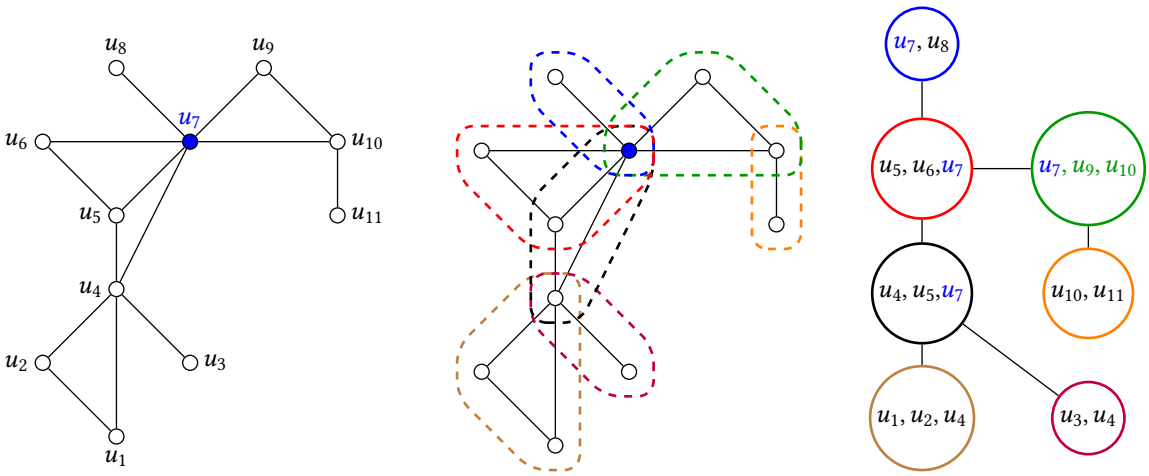
(d) Turn 6. (e) Turn 8. (f) Turn 9.

Fig. 1. An example of a feasible solution for the instance encoded in subfigure (a), for a communication constraint of $d = 1$. The colors on the agents and the vertices are used to encode the terminal vertex of each agent. Observe also that if each agent followed their optimal path to reach their terminal (the horizontal path on which they start), we would have a schedule of length 2, but this would not be a feasible solution, as the communication constraint would not be respected during the first and second turns.

(2) for all $i \in [\mu]$ and $a, b \in A$ where $a \neq b$, we have that $s_i(a) \neq s_i(b)$,
(3) the vertex set $\{s_i(a) \mid a \in A\}$ is d-connected, for every $i \in [\mu]$, and
(4) for every agent $a \in A$, we have $s_\mu(a) = t(a)$.

Moreover, we do not allow swaps. For ease of exposition, we will refer to the condition (3) above as the *communication constraint*. A feasible solution $s_1, \ldots, s_\mu$ has *makespan* μ. Our goal is to decide if there exists a feasible solution of makespan $\mu \leq \ell$. Fig. 1 illustrates an example of a feasible solution.

Parametrized Complexity. The parametrized complexity point of view allows us to overcome the limitations of classical measures of time (and space) complexity, by taking into account additional measures that can affect the running time of an algorithm; these additional measures are exactly what we refer to as parameters. The goal here is to construct exact algorithms that run in time $f(k) \cdot \text{poly}(n)$, where f is a computable function, n is the size of the input and k is the parameter. Algorithms with such running times are referred to as *fixed-parameter*



(a) The graph G

(b) Assigning the vertices of G to bags of size at most 3

(c) A tree-decomposition $\mathcal{T}$ of G of width 2

Fig. 2. Illustration of a tree-decomposition. Every edge of G appears in at least one node of $\mathcal{T}$. For every vertex of G (e.g. u_7), the nodes that contain that vertex form a tree of $\mathcal{T}$. This decomposition is of width 2 as its largest bag is of order 3.

tractable (FPT). A problem admitting such an algorithm belongs to the class FPT. Similar to classical complexity theory, there is also a notion of infeasibility. A problem is presumably not in FPT if it is shown to be W[1]-hard (by a parameterized reduction). We refer the interested reader to now classical monographs Cygan et al. (2015); Niedermeier (2006); Flum and Grohe (2006); Downey and Fellows (2012) for a more comprehensive introduction to this topic.

Structural Parameters and Logic. The main structural parameter that interests us in this work is that of *treewidth*.A *tree-decomposition* of G is a pair $(\mathcal{T}, \beta)$, where $\mathcal{T}$ is a tree, $\beta\colon V(\mathcal{T}) \to 2^V$ is a function assigning each node x of $\mathcal{T}$ its *bag*, and the following conditions hold:

- for every edge $\{u, v\} \in E(G)$ there is a node $x \in V(\mathcal{T})$ such that $u, v \in \beta(x)$, and
- for every vertex $v \in V$, the set of nodes x with $v \in \beta(x)$ induces a connected subtree of $\mathcal{T}$.

The *width* of a tree-decomposition $(\mathcal{T}, \beta)$ is $\max_{x \in V(\mathcal{T})} |\beta(x)| - 1$, and the treewidth $\mathrm{tw}(G)$ of a graph G is the minimum width of a tree-decomposition of G. In Figure 2 we illustrate an example of the above notions.

It is known that computing a tree-decomposition of minimum width NP-hard Arnborg et al. (1987), but it is fixed-parameter tractable when parameterized by the treewidth Kloks (1994); Bodlaender (1996), and even more efficient algorithms exist for obtaining near-optimal tree-decompositions Korhonen and Lokshtanov (2023).

In our work, we make use of the celebrated Courcelle Theorem Courcelle (1990), stating that any problem that is expressible by a monadic second-order formula can be solved in FPT-time parameterized by the treewidth of G. MSO logic is an extension of first-order logic, distinguished by the introduction of set variables (denoted by uppercase letters) that represent sets of domain elements, in contrast to individual variables (denoted by lowercase letters), which represent single elements. Specifically, we utilize MSO_2, a variant of MSO logic that allows quantification over both the vertices and edges. This extension enables us to address a broader class of problems. More generally, Courcelle's algorithm extends to the case when both the graph G and the MSO_2 language are enriched with finitely many vertex and edge labels.

3 The Problem is Very Hard

In this section, we will prove the following theorem.

Theorem 1. *The Multiagent Path Finding with Communication Constraints problem is* W*[1]-hard parameterized by the number of agents, even for* $\ell = 3$ *and* $d = 1$.

Proof. The reduction is from the k-Multicolored Clique (k-MCC for short) problem. This problem takes as input a graph $H = (V, E)$, whose vertex set V is partitioned into the k independent sets $S_1, \ldots, S_k$. The question is whether there exists a clique on k vertices as a subgraph of H. Observe that if such a clique does exist, then it contains a unique vertex from S_i, for each $i \in [k]$. This problem was shown to be W[1]-hard in Pietrzak (2003).

Starting from an input of the k-MCC problem, consisting of a graph H whose vertex set is partitioned into sets $S_1, \ldots, S_k$, we will construct an instance $I = \langle G, A, s_0, t, 1, 3\rangle$ of MAPFCC such that I is a yes-instance if and only if H contains a clique on k vertices.

The construction of G. First, we describe the two gadgets that will serve as the building blocks of G. For each $i \in [k]$, let $S_i = \{v^i_1, \ldots, v^i_n\}$; we build the V_i gadget as follows (see Fig. 3). We begin with n paths with $k-1$ vertices each, each corresponding to a vertex of S_i. So, for each $p \in [n]$, we have the path $P^i_p = v^i_{p,1}v^i_{p,2}\ldots v^i_{p,i-1}v^i_{p,i+1}\ldots v^i_{p,k}$, which excludes the vertex $v^i_{p,i}$. We then add the new path $a^i_1 \ldots a^i_{i-1}a^i_{i+1}\ldots a^i_k$ and, for each $p \in [n]$, we add the edge $a^i_j v^i_{p,j}$, for each $j \in [k] \setminus i$. We say that the vertices $v^i_{p,1}$ and $v^i_{p,k}$, for every $p \in [n]$ are the *top* and *bottom* vertices of the V_i gadget, respectively (if $i = 1$ or $i = k$ we adapt accordingly). Next, for each $l, m \in [k]$ with $l < m$, we build the $E_{l,m}$ gadget. This gadget consists of a forest of edges, each corresponding to an edge between the vertices of S_l and S_m in H. That is, there exist vertices $u^{l,m}_p$ and $u^{m,l}_q$ in $E_{l,m}$ such that $u^{l,m}_p u^{m,l}_q \in E(E_{l,m})$ if and only if there exist vertices $v^l_p \in S_l$ and $v^m_q \in S_m$ such that $v^l_p v^m_q \in E(H)$. We say that $u^{l,m}_p$ and $u^{m,l}_q$, for every $p, q \in [n]$, are the *top* and *bottom*, respectively, vertices of $E_{l,m}$. Note that $E_{l,m}$ contains at most $O(n^2)$ vertices. Moreover, since $l < m$, we have $\binom{k}{2}$ such gadgets in total. This finishes the construction of the two gadgets we use.

We are now ready to construct the graph G. We start with a copy of the V_i gadget for each $i \in [k]$ and a copy of the $E_{l,m}$ gadget for each $l, m \in [k]$ with $l < m$. For each $i \in [k-1]$, we add all the edges between the bottom vertices of V_i and the top vertices of V_{i+1}, as well as the edge $a^i_k a^{i+1}_1$. Then, we connect the V_i gadgets with the $E_{l,m}$ gadgets as follows (illustrated in Fig. 4). For every $l, m \in [k]$ with $l < m$ and $p, q \in [n]$, for every $u^{l,m}_p \in E_{l,m}$ and $u^{m,l}_q \in E_{l,m}$, we connect $u^{l,m}_p$ with $v^l_{p,m}$ and $u^{m,l}_q$ with $v^m_{q,l}$. Next, for every $l, m \in [k-1]$ with $l < m$, we add all the edges between the bottom vertices of $E_{l,m}$ and the top vertices of $E_{l,m+1}$ as well as all the edges between the bottom vertices of $E_{l,k}$ and the top vertices of $E_{l+1,l+2}$. Then we add the clique Q with the $k(k-1)$ vertices $\{t^i_j : i \in [k] \text{ and } j \in [k] \setminus \{i\}\}$, and we connect all the vertices of every $E_{l,m}$ gadget to all the vertices of Q.

To finalize the construction of the instance I, we need to specify the set of agents, as well as the functions s_0 and t and the value of ℓ. For the set of agents, let $A = \left\{\alpha^i_j : i \in [k] \text{ and } j \in [k] \setminus \{i\}\right\}$. Then, for any $i \in [k]$ and $j \in [k] \setminus \{i\}$, let $s_0(\alpha^i_j) = a^i_j$ and $t(\alpha^i_j) = t^i_j$. Finally, we set $\ell = 3$. Intuitively, these are chosen so that the agents are obliged to move towards their goals in every turn, thus excluding the possibility for them to wait at any vertex at any given turn. Since we manage to show that the problem is hard even in this situation where the behavior of the agents is "manageable", the same holds true in the more general case. This finishes the construction of the instance I.

Before we move on with the reduction, we present some important observations. Observe first that the starting position of each agent is at a distance exactly 3 from their target position. Since in I we have $\ell = 3$, it follows that any feasible solution of I will have makespan exactly 3 and will be such that $s_1(\alpha^i_j) \in P^i_p$, for some $p \in [n]$, $s_2(\alpha^i_j) \in E_{l,m}$, for some $l, m \in [k]$ with $l < m$, and $s_3(\alpha^i_j) = t^i_j$, for every $i \in [k]$ and $j \in [k] \setminus \{i\}$. Also, observe

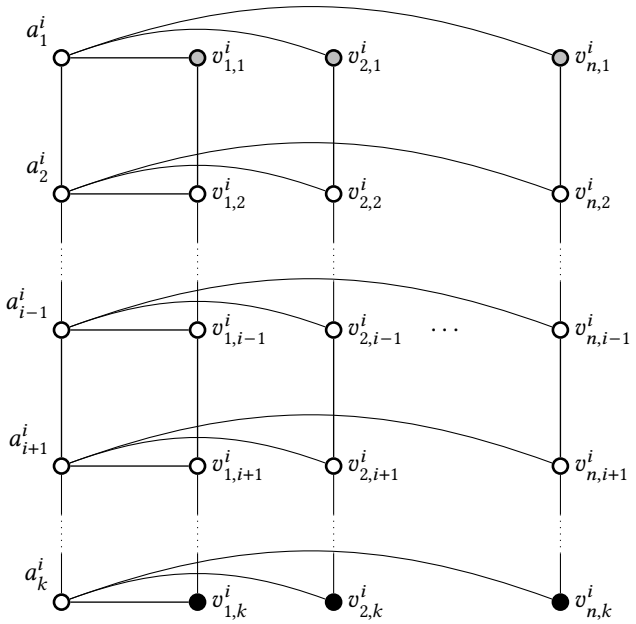


Fig. 3. The V_i gadget, for any $i \in [k]$, used in the proof of Theorem 1. The color gray (black resp.) is used to represent the top (bottom resp.) vertices of the gadget.

that in any feasible solution of I, we have that for every $i \in [k]$ there exists a unique $p \in [n]$ such that $s_1(\alpha^i_j) \in P^i_p$ for every $j \in [k] \setminus \{i\}$. In fact, assume that s_1 is such that there exist $j < j' \in [k] \setminus \{i\}$ and $p < p' \in [n]$ with $v_1 = s_1(\alpha^i_j) \in P^i_p$ and $v_2 = s_1(\alpha^i_{j'}) \in P^i_{p'}$. Then $\text{dist}_G(v_1, v_2) \geq 2$. Since $d = 1$ and I is assumed to be feasible, we have that the positions of the agents during s_1 induce a path of G that connects v_1 and v_2. By s_0 and the construction of G, this path must necessarily include a vertex a^i_j for some $j \in [k] \setminus \{i\}$; that is, there exists an $\alpha \in A$ such that $s_1(\alpha) = a^i_j$. This is a contradiction to the makespan of the solution.

The reduction. We start by assuming that I is a yes-instance of MAPFCC and let s_1, s_2, s_3 be a feasible solution of I. It follows from the previous observations that for every $i \in [k]$ there exists a unique $p \in [n]$ such that $s_1(\alpha^i_j) \in P^i_p$ for every $j \in [k] \setminus \{i\}$; let us denote this p by $p(i)$. Consider now the vertices $v^i_{p(i)}$, for every $i \in [k]$, of H. We claim that the subgraph of H that is induced by these vertices is a clique on k vertices. Indeed, assume that there are two indices $i < i' \in [k]$ such that $v^i_{p(i)} v^{i'}_{p(i')} \notin E(H)$. Consider now the vertices $v^i_{p(i),p(i')}$ and $v^{i'}_{p(i'),p(i)}$ of V_i and $V_{i'}$ respectively. By the definition of $p(i)$ and $p(i')$, we have that both of these vertices are occupied by an agent, say α and β respectively, during the turn s_1. Since s_1, s_2, s_3 is a feasible solution of I, and by the construction of G, we have that $s_2(\alpha)$ and $s_2(\beta)$ belong to $E_{i,i'}$. In particular, we have that $s_2(\alpha) = u^{i,p(i')}_{p(i)}$ and $s_2(\beta) = u^{i',p(i)}_{p(i')}$. This is a contradiction since, by its construction, the $E_{i,i'}$ gadget does not include these vertices since $v^i_{p(i)} v^{i'}_{p(i')} \notin E(H)$.

For the reverse direction, we assume that H is a yes-instance of k-MCC. That is, for each $i \in [k]$, there exists a $p(i) \in [n]$ such that the vertices $\{v^i_{p(i)} : i \in [k]\}$ form a clique of H. For each $i \in [k]$ and $j \in [k] \setminus \{i\}$:

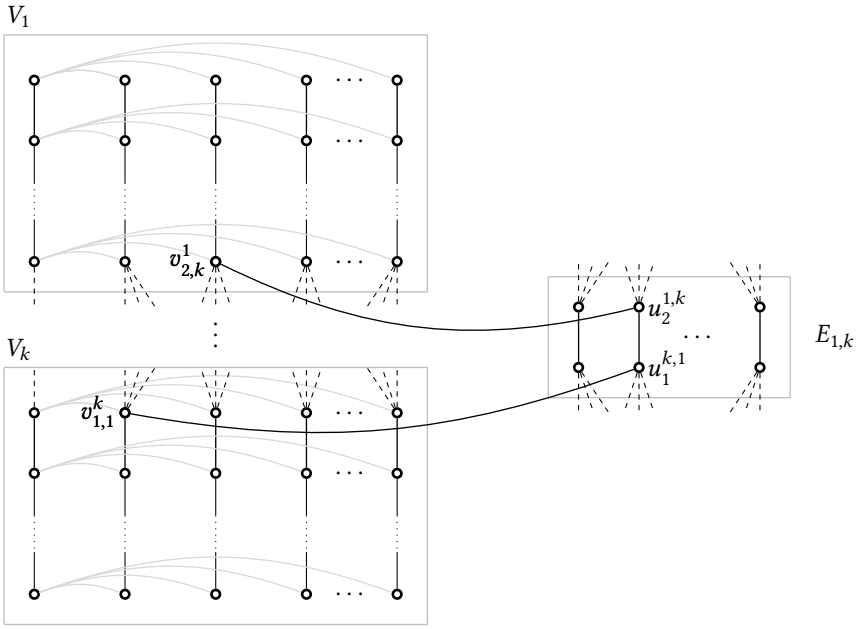


Fig. 4. An example of the connection between the V_i and the $E_{l,m}$ gadgets used in the proof of Theorem 1. The edge $u^{1,k}_2 u^{k,1}_1$ of $E_{1,k}$ represents the edge $v^1_2 v^k_1$ of H, where $v^1_2 \in S_1$ and $v^k_1 \in S_k$. The dotted edges are used to represent the edges connecting the vertices between the corresponding gadgets.

(1) We set $s_1(\alpha^i_j) = v^i_{p(i),j}$. Clearly, $s_0(\alpha^i_j)$ is a neighbor of $s_1(\alpha^i_j)$. Moreover, since $v^i_{p(i),k}$ is connected to $v^{i+1}_{p(i+1),1}$ for every $i \in [k-1]$, we have that the communication constraint is respected within s_1.
(2) Next, we set $s_2(\alpha^i_j) = u^{i,j}_{p(i)}$. Observe that for each $i \in [k]$ and $j \in [k] \setminus \{i\}$, we have that $v^i_{p(i),j}$ is a neighbor of $u^{i,j}_{p(i)}$ by the construction of G and because the vertices $\{v^i_{p(i)} : i \in [k]\}$ form a clique of H. By the same arguments, and by the connectivity between the $E_{l,m}$ gadgets, we have that the connectivity constraint is also respected within s_2.
(3) Finally, we set $s_3(\alpha^i_j) = t^i_j$. It is trivial to check that $s_3(\alpha^i_j)$ is a neighbor of $s_2(\alpha^i_j)$ and that the connectivity constraint is respected within s_3.

It follows that s_1, s_2, s_3 is a feasible solution of I. This completes the proof. □

At this point, we would like to observe the following detail in the construction of the previous proof. Consider the instance $I = \langle G, A, s_0, t, 1, 3\rangle$ that was constructed. If I is a yes-instance of MAPFCC and s_1, s_2, s_3 is a feasible solution, the agents find themselves on the vertices of the clique Q according to s_3. The crucial observation is that since Q is a clique, and all the vertices of Q are accessible from all the vertices of the $E_{l,m}$ gadgets, we are free to assign whichever vertex of Q to be the terminal of whichever agent, without affecting the feasibility of the solution. In other words, s_1, s_2, s_3 is also a feasible solution to the problem of Anonymous Multiagent Path Finding with Communication Constraints (AMAPFCC). This problem is defined exactly like MAPFCC, the difference being that in AMAPFCC there is a set of terminals $T \subseteq V$, and the solution is feasible if all the vertices of T are occupied by an agent (regardless of which agent it is) by the last turn of the schedule. This leads us to the following.

Corollary 1. *The Anonymous Multiagent Path Finding with Communication Constraints problem is* W*[1]-hard parameterized by the number of agents, even for* $\ell = 3$ *and* $d = 1$.

This corollary appears in stark opposition to the usual Anonymous Multiagent Path Finding problem (with no communication constraint), in which finding an optimal solution in regards to the makespan is achievable in polynomial time Yu and LaValle (2012); Ali and Yakovlev (2024).

4 Efficient Algorithms

In this section, we present our FPT algorithms that solve the MAPFCC problem.

4.1 Few Agents and Short Communication

Theorem 2. *The Multiagent Path Finding with Communication Constraints problem is in* FPT *parameterized by the number of agents* k *plus the maximum degree* Δ *and the communication range* d.

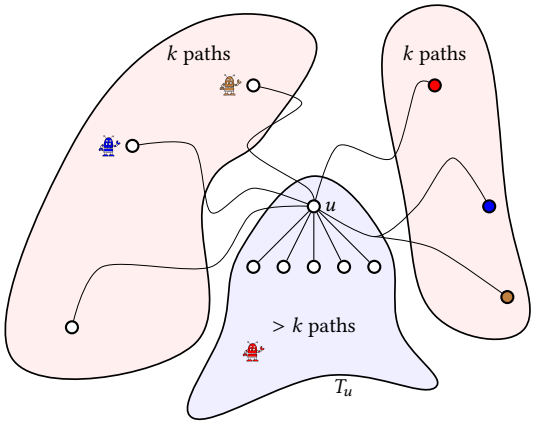
Proof. Let $\langle G, A, s_0, t, d, \ell\rangle$ be an instance of Multiagent Path Finding with Communication Constraints. The algorithm is as follows. We build an auxiliary directed graph H which has a vertex u for every possible arrangement of the k agents of A into feasible (d-connected) positions. Observe that $V(H)$ contains the vertices u_s and u_t which correspond to the initial and the final configurations of the agents on the vertices of G, respectively. Two vertices $u_1, u_2 \in V(H)$ are joined by the arc (u_1, u_2) if and only if it is possible to move from the configuration represented by the vertex u_1 into the one represented by u_2 in one turn. Clearly, $\langle G, A, s_0, t, d, \ell\rangle$ is a yes-instance of Multiagent Path Finding with Communication Constraints if and only if there is a directed path in H from u_s to u_t, of length at most ℓ. That is, if $\mathrm{dist}_H(u_s, u_t) \leq \ell$; which one can check, e.g., using BFS. We will show that $|V(H)| \leq \Delta^{O(dk)} kd^k n$, which suffices to prove the statement.

Let us consider an agent $a \in A$. We will first count the different feasible arrangements of the k agents of A (i.e., the possible positions of the agents on the graph) such that a is located at a vertex $u \in V(G)$. Since two agents are considered connected if they are in distance at most d and we have k agents, there must be a set U of kd vertices such that the induced subgraph $G[U]$ is connected and all agents are located in U. It is known (see (Guillemot and Marx, 2014, Proposition 5.1)) that given u, there exist at most $\Delta^{O(dk)}$ sets U of size dk where $u \in U$ and $G[U]$ is connected. We can also enumerate all such sets U in $\Delta^{O(dk)}$ time. Finally, we need to know the exact positions of the agents in U. Since the possible arrangement of the agents in U are $\binom{kd}{k}k! \leq kd^k$, we have that $|V(H)| \leq \Delta^{O(dk)} kd^k n$. □

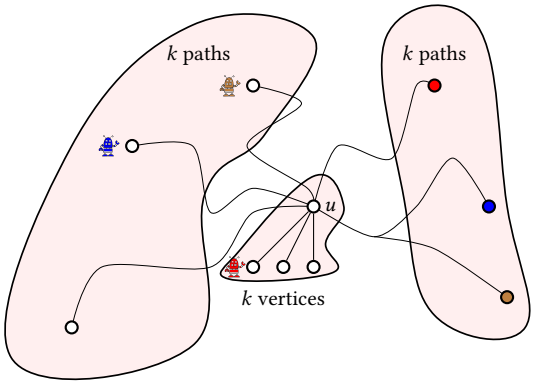
In the next theorem, we use the algorithm presented in Theorem 2 in order to efficiently solve the MAPFCC problem on trees when we have a bounded number of agents and communication range. On a high level, the idea is that if the tree has a vertex u of very high degree, then it will also contain a lot of "useless" vertices neighboring u. Indeed, at any given time, the pertinent neighbors of u are at most equal to the number of agents, which is bounded. Once we remove these "useless" vertices, we are left with a graph of bounded maximum degree, at which point we may employ the previous algorithm. In the upcoming proof we formalize this idea.

Theorem 3. *The Multiagent Path Finding with Communication Constraints problem is in* FPT *parameterized by the number of agents* k *plus the communication range* d *when the input graph is a tree.*

Proof. Let $I = \langle T, A, s_0, t, d, \ell\rangle$ be an instance of Multiagent Path Finding with Communication Constraints, where T is a tree. The idea here is to create a new instance $I' = \langle T', A, s_0, t, d, \ell\rangle$ such that I is a yes-instance if and only if the same is true for I', where T' is a tree of maximum degree $3k$. In essence, we will prune the tree T so that its maximum degree becomes bounded by $3k$. Furthermore, we show that, given a valid schedule for I, we can adapt the movements of the agents appropriately to obtain a valid schedule for I' of the



(a) The tree T (unpruned). The paths connecting the vertices in the left (right resp.) blob to u correspond to $P^-(u)$ ($P^+(u)$ resp.). The lavender blob corresponds to the tree T_u.



(b) The tree $\hat{T}$ (pruned). The red agent is placed at one of the k remaining neighbors of u for the same number of turns as it would spend in the pruned part.

Fig. 5. An abstract illustration of the trees T, T_u and $\hat{T}$ employed in the proof of Theorem 3. The colors on the agents and the vertices are used to encode the terminal vertex of each agent. In each subfigure, the left (right resp.) blob contains the starting (terminal resp.) positions of the agents. Both subfigures depict the instance during the same turn i, when the red agent has moved to some vertex inside T_u in T and some neighbor of u in $\hat{T}$.

same length. Once this is done, it suffices to apply the algorithm provided in Theorem 2 to decide whether I' is a yes-instance or not.

We now describe the pruning procedure (see Figure 5). If $\Delta(T) \leq 3k$, then we are done. So, let $u \in V(T)$ be such that $d_T(u) > 3k$. For each $j \in [k]$, we define $P^-_{j,u}$ as the simple path of T that connects $s_0(a_j)$ to u. Similarly, let $P^+_{j,u}$ be the simple path of T that connects u to $t(a_j)$. Observe that in the case where $u = s_0(a_j)$ ($u = t(a_j)$ respectively), for some $j \in [k]$, then $P^-_{j,u} = \emptyset$ ($P^+_{j,u} = \emptyset$ respectively). Then, we define $P^-(u) = \{P^-_{1,u}, \ldots, P^-_{k,u}\}$ and $P^+(u) = \{P^+_{1,u}, \ldots, P^+_{k,u}\}$. Intuitively, the set $P^-(u)$ contains all the paths of T that will be relevant for the agents to reach u from their initial positions. The set $P^+(u)$ contains all the paths of T that will be relevant for the agents to reach their targets from u. Finally, let $V_u = N(u) \cap (P^-(u) \cup P^+(u))$. That is, V_u contains the neighbors of u that are relevant with respect to the paths mentioned above. Observe that $|V_u| \leq 2k$. Let T_u be the subtree of $T[V \setminus V_u]$ that contains u. Since $d_T(u) > 3k$, we have that T_u contains at least the vertices $N(u) \setminus V_u$. We are now ready to describe the pruning that we perform: starting from T, remove all the vertices of T_u, except from u and k of its neighbors $v_1, \ldots, v_k$ in T_u; let $\hat{T}$ be the resulting graph.

Claim 1. *If $I = \langle T, A, s_0, t, d, \ell \rangle$ is a yes-instance, then $\hat{I} = \langle \hat{T}, A, s_0, t, d, \ell \rangle$ is also a yes-instance.*

Proof of the claim. Assume that $s = (s_1, \ldots, s_\ell)$ is a feasible solution of I; we construct a feasible solution $s' = s'_1, \ldots, s'_\ell$ of I'. We start by setting $s'_0(a) = s_0(a)$ for every $a \in A$. Then, for every $i \in [\ell]$ and for every $j \in [k]$, we set

$$s'_i(a_j) = \begin{cases} v_j, & \text{if } s_j(a_j) \in T_u \setminus u \\ s_i(a_j), & \text{otherwise.} \end{cases}$$

First, we need to show that $s'_i(a)$ is a neighbor of $s'_{i-1}(a)$ for every $a \in A$. Let $a \in A$ and $i \in [\ell]$. We distinguish the following cases:

$s_i(a) \in T_u \setminus u$ and $s_{i-1}(a) \in T_u \setminus u$ Then $s'_i(a) = s'_{i-1}(a) = v_j$ for some $j \in [k]$.
$s_i(a) \in T_u \setminus u$ and $s_{i-1}(a) \notin T_u \setminus u$ Since s is a feasible solution, we have that $s_{i-1}(a) = u$. Thus, $s'_i(a) = v_j$ for some $j \in [k]$ and $s'_{i-1}(a) = s_{i-1}(a) = u$.
$s_i(a) \notin T_u \setminus u$ and $s_{i-1}(a) \in T_u \setminus u$ Is analogous to the previous one.
$s_i(a) \notin T_u \setminus u$ and $s_{i-1}(a) \notin T_u \setminus u$ Then $s'_i(a) = s_i(a)$ and $s'_{i-1}(a) = s_{i-1}(a)$ and the feasibility of s' is guaranteed by the feasibility of s.

Next, we need to show that $s'_\ell(a_j) = t(a_j)$ for every $j \in [k]$. This follows directly from the definitions of s' and the paths $P^-_{j,u}$ and $P^+_{j,u}$ and the fact that s is a feasible schedule.

Finally, we will ensure that the connectivity constraint is preserved in s'. Let D and D' be the communication graphs of T and $\hat{T}$ respectively. We will show that $D'[\{s'_i(a) \mid a \in A\}]$ is connected for every $i \in [\ell]$. Assume that this is not true, and let $l \in [\ell]$ be one turn during which the connectivity constraint fails in s'. That is, $D'[\{s'_l(a) \mid a \in A\}]$ contains at least two connected components. Since s is a feasible solution of I, it follows that there exist two agents b and c such that $\text{dist}_T(s_l(b), s_l(c)) \leq d$ but $\text{dist}_{\hat{T}}(s'_l(b), s'_l(c)) > d$. By the definition of s', we have that at least one of the agents b and c is located in $T_u \setminus u$ according to s'_l. We distinguish the following cases for the values of $d \geq 2$ (we deal with the case where $d = 1$ afterwards):

$s'_l(b) \in T_u \setminus u$ and $s'_l(c) \notin T_u \setminus u$ It holds that $\text{dist}_{T'}(s'_l(b), s'_l(c)) \leq \text{dist}_T(s_l(b), s_l(c)) - 1 \leq d - 1$.
$s'_l(b) \in T_u \setminus u$ and $s'_l(c) \in T_u \setminus u$ In this case, there exist two vertices v_x and v_z, for some $x < z \in [k]$ which are leafs attached to u in T', such that $s'_l(b) = v_x$ and $s'_l(c) = v_z$. Clearly, $\text{dist}(v_x, v_z) = 2 \leq d$.

Lastly, we deal with the particular case of $d = 1$. We additionally assume that ℓ is optimal, that is, there is no shorter feasible schedule. We claim that there exists an agent $a^* \in A$ such that $s'_l(a^*) = s_l(a^*) = u$. Let us assume that this is not true. We distinguish the following two sub-cases:

- The agent b is such that $s_l(b) = s'_l(b) = x \in V(P^-(u)) \cup V(P^+(u))$. In this case, and since at least one of the agents b and c must be in $T_u \setminus u$ during the turn l, the existence of a^* is guaranteed by the feasibility of I for $d = 1$.
- Every agent $a \in A$ is such that $s_l(a) \in T_u$. Consider $l_0 < l$, the last turn that the vertex u was occupied before the turn l, say by the agent e. Also, let $l < l_1 \leq \ell$ be the first turn that the vertex u will be occupied after the turn l, say by the agent g. Let us also consider what happens in T' according to s' during these turns. We have that at the turn l_0 all the agents of A are on the leafs attached to u except for the agent e which is on u. Then, on the turn $l_0 + 1$, all the agents of A are on the leafs attached to u, and remain there until the turn l_1, where the agent g moves to u. We then modify s' by setting $s'_{l_0+1}(u) = g$. If $l_0 + 1 = l$, we have a contradiction as u is assumed to be empty during the turn l according to s'. Thus, $l_0 + 1 < l$. But in this case, the makespan of the modified s' is smaller than ℓ (note that s' is a solution for I as well). But ℓ was assumed to be optimal, leading to a contradiction.

In all the cases above we are lead to a contradiction, proving that the connectivity constraint of s is indeed preserved by s'. ◇

For the other direction of the equivalence, it holds trivially since $\hat{T}$ is a subgraph of T. Therefore, we apply the above pruning procedure exhaustively, that is, as long as there is a vertex of degree at least $3k + 1$. In this way, we obtain the tree T' and the instance I'. □

4.2 Tree-like Structures

In the previous section we presented an efficient algorithm to solve MAPFCC when we have a bounded number of agents, communication distance and maximum degree. Furthermore, we can relax the requirement for a bounded maximum degree if we consider graphs that are trees. The next natural question to pose is whether this remains true when we consider graphs that are "close to being trees", i.e., have bounded treewidth. In the next theorem

we answer positively to this question. This is done by capturing the dynamicity of the problem through a static auxiliary graph (the graph G_I, see Figure 6 for an example). We then prove that if the original graph has bounded treewidth, the same holds true for the auxiliary graph. Finally, we encode the MAPFCC problem on the auxiliary graph in an MSO_2 formula, and employ Courcelle's theorem Courcelle (1990) to construct an FPT algorithm.

Theorem 4. *The MAPFCC problem is in* FPT *parameterized by the treewidth* tw *of G plus the makespan ℓ and the communication range d.*

Proof. Let $I = \langle G, A, s_0, t, d, \ell \rangle$ be an instance of MAPFCC. Our goal is to construct an auxiliary graph G_I with special vertex and edge labels, such that (i) the treewidth of G_I is at most $3\ell \cdot \text{tw}$ whenever I is a yes-instance, and (ii) the existence of a solution can be expressed by an MSO_2 sentence over G_I. The claim then follows by testing the treewidth of G_I followed by a standard use of Courcelle's theorem Courcelle (1990).

Let $G = (V, E)$ be the graph of the input instance. We start by constructing a labeled auxiliary graph G_I (see Figure 6). Before we do this, allow us to provide a short intuition on the graph G_I. In essence, our goal with G_I is to capture the dynamicity of the problem. This is done by considering consecutive "snapshots" of G. These snapshots are then linked by considering which vertices are available to an agent in the next turn, according to its current position. Finally, for each agent a, we add an edge between the starting position of a in the first snapshot and the terminal vertex of a in the last snapshot. This is done to encode the information of where does each agent want to go into the bags of the tree-decomposition of G_I.

We now proceed with formally defining G_I. Its vertex set is composed of sets $V_0, \ldots, V_\ell$ where V_i contains one copy of each vertex of V, that is, $V_i = \{v_i \mid v \in V\}$ (we denote by v_i the copy of the vertex v in V_i). We refer to these sets as *layers* and we give all the vertices in V_i a vertex label vertex_i. The graph G_I contains four different types of edges with four distinct edge labels defined as follows.

(1) For every $v \in V$ and $i \in [\ell]$, we add to G_I the edge $v_{i-1}v_i$ with an edge label copy.
(2) For every $uv \in E$ and $i \in \{0, \ldots, \ell\}$, we add to G_I the edge $u_i v_i$ with an edge label communication.
(3) For every $uv \in E$ and $i \in [\ell]$, we add to G_I the edges $u_{i-1}v_i$ and $v_{i-1}u_i$ with an edge label cross.
(4) Finally, for every agent $a \in A$, we add to G_I the edge $s_0(a)_0 t(a)_\ell$ with an edge label agent.

Observe that the first three types of edges correspond exactly to the strong product of G with a path of length ℓ. In a sense, the construction combines the time-expanded graphs used previously for MAPF Fioravantes et al. (2024) with the augmented graphs considered for edge-disjoint paths Zhou et al. (2000); Ganian et al. (2021). Importantly, we observe that the existence of a solution is equivalent to the existence of a set of vertex-disjoint paths in G_I with some additional properties.

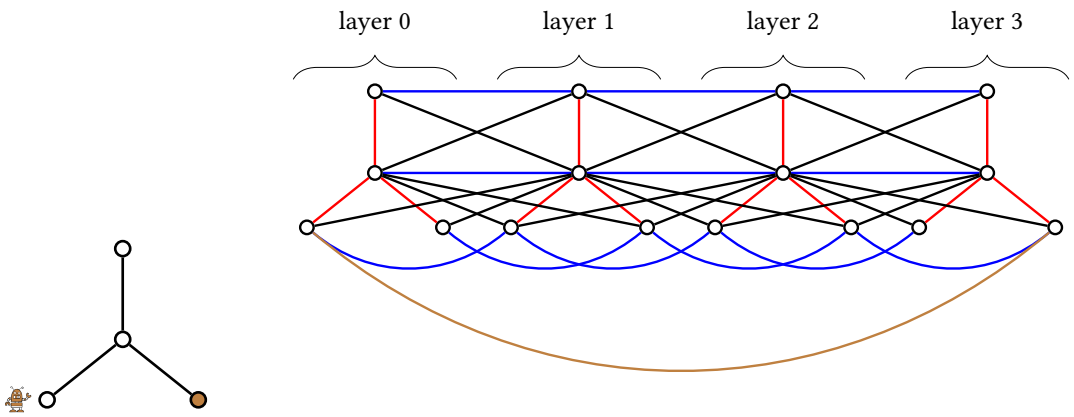
Observation 1. *The instance I is a yes-instance if and only if there exists a set of vertex-disjoint paths $\mathcal{P} = \{P_a \mid a \in A\}$ such that:*

(1) *each P_a contains exactly one vertex from each layer,*
(2) *the endpoints of each P_a are the vertices $s_0(a)_0$ and $t(a)_\ell$,*
(3) *there are no two paths P_a and P_b, vertices $u, v \in V$ and $i \in [\ell]$ such that P_a contains the edge $u_{i-1}v_i$ and P_b contains the edge $v_{i-1}u_i$, and*
(4) *for each $i \in \{0, \ldots, \ell\}$, the set of vertices $W_i \subseteq V_i$ visited by paths from $\mathcal{P}$ forms a d-connected set in $G[V_i]$.*

We show later how to translate these properties into MSO_2 predicates. Unfortunately, it is not guaranteed that G_I must have a small treewidth due to the edges that connect the targets and destinations of the individual agents. However, we can bound its treewidth whenever I is a yes-instance.

Claim 2. *If I is a yes-instance, then the treewidth of G_I is at most $O(\ell \cdot \text{tw})$, where* tw *is the treewidth of G.*

Proof of the claim. Let (T, β) be a tree-decomposition of G of optimal width tw. First, we consider a graph G'_I obtained from G_I by considering only the edges with labels copy, communication and cross. We define its



(a) The graph G. (b) The graph G_I.

Fig. 6. An example of a graph G and the corresponding graph G_I for $\ell = 3$, used in the proof of Theorem 4. In G, the brown agent is set next to that agents' staring vertex and the brown vertex is its target. The colors on the edges of G_I are used to represent the various labels we employ. In particular, blue edges are labeled copy, red edges are labeled communication, black edges are labeled cross and the brown edge is labeled agent.

tree-decomposition (T, β') by replacing every occurrence of a vertex v in any bag with its $\ell + 1$ copies $v_0, \ldots, v_\ell$. It is easy to see that this is a valid tree-decomposition of G'_I of width $O(\ell \cdot \mathrm{tw})$.

Now, we define a tree-decomposition (T, β'') of G_I assuming that I is a yes-instance. By Observation 1, there is a set of vertex-disjoint paths $\{P_a \mid a \in A\}$ connecting the terminals of each agent. We obtain β'' from β' by adding the vertices $s_0(a)_0$ and $t(a)_\ell$ to every bag intersected by the path P_a for each $a \in A$. Observe that for every vertex $v \in G_I$, the set of nodes x with $v \in \beta(x)$ remains connected and, moreover, we guarantee that $s_0(a)_0$ and $t(a)_\ell$ appear together in some bag, for example, a bag that originally contained only one of the terminals.

Finally, let us bound the width of the tree-decomposition (T, β''). Since every vertex can lie on at most one path P_a for some $a \in A$, we added to $\beta''(x)$ at most two new vertices for every vertex $v \in \beta'(x)$. Therefore, $\beta''(x)$ contains at most $3 \cdot |\beta'(x)|$ vertices and the tree-decomposition (T, β'') has width $O(\ell \cdot \mathrm{tw})$ as promised. ◇

Now, we show how to encode the existence of a feasible solution into an MSO_2 sentence. Formally, we construct MSO sentence over the signature consisting of a single binary relation symbol inc that verifies the incidence between a given vertex and edge, and unary relation symbols agent, copy, cross, communication and $\mathsf{vertex}_0, \ldots, \mathsf{vertex}_\ell$ that exactly correspond to the respective edge and vertex labels. We proceed in two steps. First, we translate Observation 1 into an existential statement about edge and vertex sets in G_I with special properties expressed only in terms of the labels. Afterwards, we show how to encode it in MSO_2.

Claim 3. *The instance I is a yes-instance if and only if there exists a set of edges S and sets of vertices $X_0, \ldots X_\ell$ in G_I such that*

(1) *all vertices in X_i are labeled* vertex_i *for every $i \in \{0, \ldots, \ell\}$,*
(2) *all edges in S are labeled* copy *or* cross,
(3) *every vertex in X_i for $i \in [\ell - 1]$ is incident to exactly two edges from S that connect it to vertices in X_{i-1} and X_{i+1},*
(4) *every vertex in X_0 and X_ℓ is incident to exactly one edge from S,*
(5) *every vertex outside of $X_0 \cup \cdots \cup X_\ell$ is not incident to any edge from S,*

(6) *for every edge e with label* agent*, there is a subset $T \subseteq S$ such that the endpoints of e are incident to exactly one edge in T and every other vertex is incident to either zero or two edges from T,*
(7) *there are no two edges $u_1v_1, u_2v_2 \in S$ such that edges u_1v_2 and u_2v_1 both exist and are labeled* copy*, and*
(8) *X_i forms a d-connected set with respect to the edges labeled* communication *for each $i \in \{0, \dots, \ell\}$.*

*Proof of the claim.*First, let us assume that I is a yes-instance and let $\{P_a \mid a \in A\}$ be the set of vertex-disjoint paths guaranteed by Observation 1. It is straightforward to check that all properties (1)–(8) are satisfied when S consists of all edges contained in these paths and X_i for each i consists of all the vertices in the layer V_i contained in some path P_a.

Now let us assume that there exist sets S and $X_0, \dots, X_\ell$ that satisfy (1)–(8). The properties (1)–(5) guarantee that S forms the edge set of a set of vertex-disjoint paths $\mathcal{P}$ where each path $P \in \mathcal{P}$ contains exactly one vertex from each layer with endpoints in layers V_0 and V_ℓ. Therefore, the set of paths $\mathcal{P}$ satisfies condition (1) of Observation 1. Moreover, the set X_i contains exactly the vertices from the layer V_i that lie on some path $P \in \mathcal{P}$. Property (6) verifies that $\mathcal{P}$ contains a path P_a connecting the vertices $s_0(a)_0$ and $t(a)_\ell$ for each $a \in A$, i.e., condition 2 of Observation 1. Property (7) is equivalent to condition (3) of Observation 1 as it enforces that two agents cannot swap their positions along a single edge. And finally, property (8) enforces the connectivity requirement of MAPFCC equivalently to condition 4 of Observation 1. ◇

We proceed to construct an MSO_2 sentence φ equivalent to Claim 3. Its general form is

$$\varphi := \exists S, X_0, \dots, X_\ell \left(\bigwedge_{i=1}^{i} \varphi_i(S, X_0, \dots, X_\ell) \right),$$

where each φ_i is an MSO_2 formula with $\ell + 2$ free variables $S, X_0, \dots, X_\ell$ that encodes the property i of Claim 3.

In order to simplify the upcoming definitions, we define a predicate $\deg_k(v, F)$ expressing that vertex v is incident with exactly k edges from F for $k = 0, 1, 2$. We define:

$$\deg_0(v, F) := \nexists e \big(e \in F \wedge \mathrm{inc}(v, e)\big),$$
$$\deg_1(v, F) := \exists e \in F \left(\mathrm{inc}(v, e) \wedge \forall f \in F \Big(\big(\mathrm{inc}(v, f)\big) \rightarrow f = e\Big)\right),$$
$$\deg_2(v, F) := \exists e_1, e_2 \in F \left(\big(\mathrm{inc}(v, e_1) \wedge \mathrm{inc}(v, e_2)\big) \wedge e_1 \neq e_2 \wedge \forall f \in F \Big(\big(\mathrm{inc}(v, f)\big) \rightarrow (f = e_1 \vee f = e_2)\Big)\right).$$

Additionally, we use the predicate '$e = uv$' to express that edge e joins vertices u and v. Formally, it is a syntactic shorthand for $\mathrm{inc}(u, e) \wedge \mathrm{inc}(v, e) \wedge u \neq v$.

The encoding of properties (1)–(5) in MSO_2 is now fairly straightforward albeit technical

$$\varphi_1(S, X_0, \dots, X_\ell) := \bigwedge_{i=0}^{\ell} \forall v \big(v \in X_i \rightarrow \mathsf{vertex}_i(v)\big),$$
$$\varphi_2(S, X_0, \dots, X_\ell) := \forall e \Big(e \in S \rightarrow \big(\mathsf{copy}(e) \vee \mathsf{cross}(e)\big)\Big),$$
$$\varphi_3(S, X_0, \dots, X_\ell) := \bigwedge_{i=1}^{\ell-1} \left(\forall v \left(v \in X_i \rightarrow \exists u, w, e, f \begin{pmatrix} \deg_2(v, S) \wedge u \in X_{i-1} \wedge \ w \in X_{i+1} \\ \wedge\ e, f \in S \wedge e = uv \wedge f = vw \end{pmatrix}\right)\right),$$
$$\varphi_4(S, X_0, \dots, X_\ell) := \forall v \big((v \in X_0 \vee v \in X_\ell) \rightarrow \deg_1(v, S)\big),$$
$$\varphi_5(S, X_0, \dots, X_\ell) := \forall v \left(\left(\bigwedge_{i=0}^{\ell} v \notin X_i\right) \rightarrow \deg_0(v, S)\right).$$

The encoding of property (6) is cumbersome but nevertheless still straightforward as

$$\varphi_6(S, X_0, \ldots, X_\ell) \coloneqq \forall e \left(\mathsf{agent}(e) \rightarrow \exists\, u, v, T \left(\begin{array}{c} T \subseteq S \wedge e = uv \wedge \deg_1(u, T) \wedge \deg_1(v, T) \\ \wedge\ \forall w \left(\deg_0(w, T) \vee \deg_2(w, T) \vee w = v \vee w = u \right) \end{array} \right) \right).$$

Property (7) is expressed readily as

$$\varphi_7(S, X_0, \ldots, X_\ell) \coloneqq \nexists u_1, v_1, u_2, v_2, e_1, e_2, f_1, f_2 \begin{pmatrix} e_1 = u_1v_1 \wedge e_2 = u_2v_2 \wedge f_1 = u_1v_2 \wedge f_2 = u_2v_1 \\ \wedge\ e_1, e_2 \in S \wedge \mathsf{copy}(f_1) \wedge \mathsf{copy}(f_2) \end{pmatrix}.$$

In order to express the last property, we first need to encode the distance with respect to the edges with the label communication. Specifically, we define the predicate $\mathrm{dist}_k(u, v)$ that encodes that this distance between u and v is at most k. We define $\mathrm{dist}_0(u, v) \coloneqq (u = v)$ and we define $\mathrm{dist}_k(u, v)$ for $k > 0$ inductively as

$$\mathrm{dist}_k(u, v) \coloneqq \mathrm{dist}_{k-1}(u, v) \vee \exists w, e \Big(\mathrm{dist}_{k-1}(u, w) \wedge \mathsf{inner}(e) \wedge e = wv \Big).$$

We now wish to define a predicate $\mathit{connected}_k(X)$ verifying that a vertex set X is k-connected with respect to the communication edges. However, it is easier to express that X is not k-connected. That happens if and only if we can partition X into two sets A and B such that for any pair of vertices $u \in A$ and $v \in B$, their distance with respect to the communication edges is strictly more than k. Hence, we define

$$\mathit{connected}_k(X) \coloneqq \nexists A, B \begin{pmatrix} \forall v\ (v \in X \rightarrow ((v \in A \vee v \in B)\ \wedge \neg(v \in A \wedge v \in B))) \\ \wedge\ \forall u, v((u \in A \wedge v \in B) \rightarrow \neg\, \mathrm{dist}_k(u, v)) \end{pmatrix},$$

where the second line verifies that A, B form a partition of X. The definition of $\varphi_8(S, X_0, \ldots, X_\ell)$ is immediate

$$\varphi_8(S, X_0, \ldots, X_\ell) \coloneqq \bigwedge_{i=0}^{\ell} \mathit{connected}_d(X_i).$$

Full algorithm. The algorithm first computes the treewidth tw of G in $2^{O(\mathrm{tw}^3)}n$ time Bodlaender (1996). Afterwards, it constructs the graph G_I and checks whether its treewidth is within the bound given by Claim 2 using the same algorithm as in the first step. If not, then it immediately outputs a negative answer. Otherwise, it uses the celebrated Courcelle's theorem Courcelle (1990) to evaluate φ on the obtained tree-decomposition of G_I and outputs its answer. The correctness follows from Claims 2 and 3. □

Interestingly, the d-connectivity requirement can be replaced in Theorem 4 by any property definable in MSO_2, e.g., independent or dominating set. In other words, we can put a wide variety of requirements on the positions of agents in each step, and we obtain an effective algorithm parameterized by treewidth plus makespan for any such setting. Note that this closely resembles reconfiguration problems under the parallel token sliding rule Bartier et al. (2023, 2021); Kristan and Svoboda (2023).The input of such problem is a graph G with two designated vertex sets S and T of the same size and the question is whether we can move tokens initially placed on S to T by moving in each step an arbitrary subset of tokens to their neighbors where (i) there can be at most one token at a vertex at a time, (ii) tokens cannot swap along an edge, and (iii) all intermediate positions of tokens satisfy a given condition, e.g., being an independent set. In a timed version of such problem, we are additionally given an upper bound on the makespan ℓ and ask whether the reconfiguration can be carried out in at most ℓ steps.

We can naturally view the agents as tokens moving on a graph from an initial to a final set of positions using the very same reconfiguration rule. However, the tokens are now labeled, as each single agent has its required target position. From this point of view, the MAPFCC problem can be seen as a timed labeled d-connected set reconfiguration problem.

Recently, Mouawad et al. (2014) introduced a metatheorem for MSO_2-definable timed reconfiguration problems under a different reconfiguration rule. The machinery of Theorem 4 can easily be adapted to a metatheorem for MSO_2-definable timed labeled reconfiguration problems under the parallel token sliding rule.

Importantly, Theorem 4 also implies an efficient algorithm parameterized by the number of agents plus the makespan and the communication range in planar graphs and more generally in any class of graphs with bounded local treewidth. We say that a graph class $\mathcal{G}$ has *bounded local treewidth* if there is a function $f\colon \mathbb{N} \to \mathbb{N}$ such that for every graph $G \in \mathcal{G}$, its every vertex v, and every positive integer i the treewidth of $G[N^i_G[v]]$ is at most $f(i)$ where $N^i_G[v]$ is the set of all vertices in distance at most i from v. Typical examples of graph classes with bounded local treewidth are planar graphs, graphs of bounded genus, and graphs of bounded max degree.

Corollary 2. *The MAPFCC problem is in* FPT *parameterized by the number of agents k plus the makespan ℓ and the communication range d in any graph class of bounded local treewidth.*

Proof. Let $\mathcal{G}$ be a class of bounded local treewidth and let $\langle G, A, s_0, t, d, \ell\rangle$ be an instance of MAPFCC where $G \in \mathcal{G}$. Pick an arbitrary agent $a \in A$ and set $G' = G[N^{kd+\ell}_G[s_0(a)]]$, i.e., the subgraph induced by vertices in distance at most $kd + \ell$ from $s_0(a)$. Due to the connectivity requirement, all agents must start within distance at most kd from $s_0(a)$ and therefore, they cannot escape G' within ℓ steps. Moreover, the treewidth of G' is at most $f(kd + \ell)$ for some function f depending only on $\mathcal{G}$. It remains to invoke the algorithm of Theorem 4. □

Since planar graphs have bounded local treewidth Eppstein (2000), we directly get the following result.

Corollary 3. *The MAPFCC problem is in* FPT *parameterized by the number of agents k plus the makespan ℓ and the communication range d if the input is a planar graph.*

5 Conclusion

In this paper, we initiated the study of the parameterized complexity of the Multiagent Path Finding with Communication Constraints problem. Our first contribution is that even in very restricted settings, where we have a small number of agents that are obliged to move as a cohesive entity, deciding if the instance has a makespan of at most 3 is infeasible (Theorem 1). We then identify other restrictions on the input that lead to efficient algorithms (Theorems 2 through 4). This opens multiple new research directions that can be explored. First and foremost is the question of checking the efficiency of our algorithms in practice. In particular, the MSO encoding we provide for Theorem 4 is implementable by employing any state-of-the-art MSO solver (e.g., Langer (2013); Bannach and Berndt (2019); Hecher (2023)). On the other hand, one could also argue that our work provides ample motivation to follow a more heuristic approach. Even in this case, it is worth checking if our exact algorithms can be used as subroutines to improve the effective running time of the state-of-the-art algorithm that is used in practice. Finally, there is the interesting phenomenon exhibited in Corollary 1, namely that the anonymous version of the problem studied here remains computationally infeasible. It would be interesting to investigate this anonymous version further, both from the theoretical and a practical point of view. We consider all of the above important enough to warrant their respective dedicated studies.

Acknowledgments

This work was co-funded by the European Union under the project Robotics and advanced industrial production (reg. no. CZ.02.01.01/00/22_008/0004590). JMK was additionally supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS23/205/OHK3/3T/18. FF and NM acknowledge the support by the CTU Global postdoc fellowship program. DK, JMK, and MO acknowledge the support of the Czech Science Foundation Grant No. 22-19557S.

A preliminary version of our work was presented at AAAI'25 Fioravantes et al. (2025).

References

Zain Alabedeen Ali and Konstantin S. Yakovlev. 2024. Improved Anonymous Multi-Agent Path Finding Algorithm. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan (Eds.). AAAI Press, 17291–17298. https://doi.org/10.1609/aaai.v38i16.29676

Francesco Amigoni, Jacopo Banfi, and Nicola Basilico. 2017. Multirobot exploration of communication-restricted environments: A survey. *IEEE Intelligent Systems* 32, 6 (2017), 48–57.

Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. 1987. Complexity of Finding Embeddings in a k-Tree. *SIAM Journal on Algebraic Discrete Methods* 8, 2 (1987), 277–284. https://doi.org/10.1137/0608024

Max Bannach and Sebastian Berndt. 2019. Practical Access to Dynamic Programming on Tree Decompositions. *Algorithms* 12, 8 (2019), 172. https://doi.org/10.3390/A12080172

Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *International Symposium on Combinatorial Search*. 19–27.

Roman Barták, Neng-Fa Zhou, Roni Stern, Eli Boyarski, and Pavel Surynek. 2017. Modeling and Solving the Multi-agent Pathfinding Problem in Picat. In *29th IEEE International Conference on Tools with Artificial Intelligence*. 959–966. https://doi.org/10.1109/ICTAI.2017.00147

Valentin Bartier, Nicolas Bousquet, Clément Dallard, Kyle Lomer, and Amer E. Mouawad. 2021. On Girth and the Parameterized Complexity of Token Sliding and Token Jumping. *Algorithmica* 83, 9 (2021), 2914–2951. https://doi.org/10.1007/S00453-021-00848-1

Valentin Bartier, Nicolas Bousquet, and Amer E. Mouawad. 2023. Galactic token sliding. *J. Comput. Syst. Sci.* 136 (2023), 220–248. https://doi.org/10.1016/J.JCSS.2023.03.008

Hans L. Bodlaender. 1996. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM J. Comput.* 25, 6 (1996), 1305–1317. https://doi.org/10.1137/S0097539793251219

Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, Oded Betzalel, David Tolpin, and Solomon Eyal Shimony. 2015. ICBS: The Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding. In *8th Annual Symposium on Combinatorial Search*. 223–225. https://doi.org/10.1609/SOCS.V6I1.18343

Isseïnie Calviac, Ocan Sankur, and François Schwarzentruber. 2023. Improved Complexity Results and an Efficient Solution for Connected Multi-Agent Path Finding. In *22nd International Conference on Autonomous Agents and Multiagent Systems*. 1–9.

Bruno Courcelle. 1990. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Information and Computation* 85, 1 (1990), 12–75. https://doi.org/10.1016/0890-5401(90)90043-H

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

Rodney G Downey and Michael Ralph Fellows. 2012. *Parameterized complexity*. Springer Science & Business Media.

Eduard Eiben, Robert Ganian, and Iyad Kanj. 2023. The Parameterized Complexity of Coordinated Motion Planning. In *39th International Symposium on Computational Geometry*. 28:1–28:16. https://doi.org/10.4230/LIPIcs.SoCG.2023.28

David Eppstein. 2000. Diameter and Treewidth in Minor-Closed Graph Families. *Algorithmica* 27, 3 (2000), 275–291. https://doi.org/10.1007/S004530010020

Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan R. Sturtevant, Glenn Wagner, and Pavel Surynek. 2017. Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges. In *10th International Symposium on Combinatorial Search*. 29–37. https:

//doi.org/10.1609/SOCS.V8I1.18423
Foivos Fioravantes, Dusan Knop, Jan Matyás Kristan, Nikolaos Melissinos, and Michal Opler. 2024. Exact Algorithms and Lowerbounds for Multiagent Path Finding: Power of Treelike Topology. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*. 17380–17388. https://doi.org/10.1609/AAAI.V38I16.29686
Foivos Fioravantes, Dusan Knop, Jan Matyás Kristan, Nikolaos Melissinos, and Michal Opler. 2025. Exact Algorithms for Multiagent Path Finding with Communication Constraints on Tree-Like Structures. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*. AAAI Press, 23177–23185. https://doi.org/10.1609/AAAI.V39I22.34483
Jörg Flum and Martin Grohe. 2006. *Parameterized Complexity Theory*. Springer. https://doi.org/10.1007/3-540-29953-X
Robert Ganian, Sebastian Ordyniak, and M. S. Ramanujan. 2021. On Structural Parameterizations of the Edge Disjoint Paths Problem. *Algorithmica* 83, 6 (2021), 1605–1637. https://doi.org/10.1007/S00453-020-00795-3
Oded Goldreich. 2011. *Finding the Shortest Move-Sequence in the Graph-Generalized 15-Puzzle Is NP-Hard*. Springer, Berlin, Heidelberg, 1–5. https://doi.org/10.1007/978-3-642-22670-0_1
Sylvain Guillemot and Dániel Marx. 2014. Finding small patterns in permutations in linear time. In *Proceedings of the Twenty-Fifth Annual Symposium on Discrete Algorithms, SODA 2014*. 82–101. https://doi.org/10.1137/1.9781611973402.7
Jayesh K. Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops*. 66–83.
Robert A Hearn and Erik D Demaine. 2005. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science* 343, 1-2 (2005), 72–96.
Markus Hecher. 2023. Advanced tools and methods for treewidth-based problem solving. *IT-Information Technology* 65, 1-2 (2023), 65–73. https://doi.org/10.1515/itit-2023-0004
Geoffrey A Hollinger and Sanjiv Singh. 2012. Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE Transactions on Robotics* 28, 4 (2012), 967–973.
Ton Kloks. 1994. *Treewidth, Computations and Approximations*. Lecture Notes in Computer Science, Vol. 842. Springer. https://doi.org/10.1007/BFb0045375
Richard E Korf. 1990. Real-time heuristic search. *Artificial intelligence* 42, 2-3 (1990), 189–211.
Tuukka Korhonen and Daniel Lokshtanov. 2023. An Improved Parameterized Algorithm for Treewidth. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, Barna Saha and Rocco A. Servedio (Eds.). ACM, 528–541. https://doi.org/10.1145/3564246.3585245
Daniel Kornhauser, Gary L. Miller, and Paul G. Spirakis. 1984. Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups, and Applications. In *25th Annual Symposium on Foundations of Computer Science*. 241–250. https://doi.org/10.1109/SFCS.1984.715921
Jan Matyás Kristan and Jakub Svoboda. 2023. Shortest Dominating Set Reconfiguration Under Token Sliding. In *Fundamentals of Computation Theory - 24th International Symposium, FCT 2023, Trier, Germany, September 18-21, 2023, Proceedings (Lecture Notes in Computer Science, Vol. 14292)*, Henning Fernau and Klaus Jansen (Eds.). Springer, 333–347. https://doi.org/10.1007/978-3-031-43587-4_24
Alexander Langer. 2013. *Fast algorithms for decomposable graphs*. Ph. D. Dissertation. RWTH Aachen University. http://darwin.bth.rwth-aachen.de/opus3/volltexte/2013/4531
Amer E. Mouawad, Naomi Nishimura, Venkatesh Raman, and Marcin Wrochna. 2014. Reconfiguration over Tree Decompositions. In *9th International Symposium on Parameterized and Exact Computation*. 246–257. https://doi.org/10.1007/978-3-319-13524-3_21
Rolf Niedermeier. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press. https://doi.org/10.1093/ACPROF:OSO/9780198566076.001.0001

Krzysztof Pietrzak. 2003. On the parameterized complexity of the fixed alphabet shortest common supersequence and longest common subsequence problems. *J. Comput. System Sci.* 67 (12 2003), 757–771. https://doi.org/10.1016/S0022-0000(03)00078-3

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

Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial intelligence* 195 (2013), 470–495.

David Silver. 2005. Cooperative pathfinding. In *AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. 117–122.

Jamie Snape, Stephen J. Guy, Jur van den Berg, Ming C. Lin, and Dinesh Manocha. 2012. Reciprocal Collision Avoidance and Multi-Agent Navigation for Video Games. In *AAAI Workshop on Multiagent Pathfinding (MAPF@AAAI 2012)*.

Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *12th International Symposium on Combinatorial Search*. 151–158. https://doi.org/10.1609/SOCS.V10I1.18510

Pavel Surynek. 2010. An optimization variant of multi-robot path planning is intractable. In *AAAI Conference on Artificial Intelligence*. 1261–1263.

Pavel Surynek. 2022. Problem Compilation for Multi-Agent Path Finding: a Survey. In *31st International Joint Conference on Artificial Intelligence*. 5615–5622. https://doi.org/10.24963/ijcai.2022/783 Survey Track.

Pavel Surynek, Roni Stern, Eli Boyarski, and Ariel Felner. 2022. Migrating Techniques from Search-based Multi-Agent Path Finding Solvers to SAT-based Approach. *Journal of Artificial Intelligence Research* 73 (2022), 553–618. https://doi.org/10.1613/JAIR.1.13318

Davide Tateo, Jacopo Banfi, Alessandro Riva, Francesco Amigoni, and Andrea Bonarini. 2018. Multiagent connected path planning: Pspace-completeness and how to deal with it. In *AAAI Conference on Artificial Intelligence*.

Richard M. Wilson. 1974. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B* 16 (1974), 86–96. https://doi.org/10.1016/0095-8956(74)90098-7

Jingjin Yu and Steven M. LaValle. 2012. Multi-agent Path Planning and Network Flow. In *Algorithmic Foundations of Robotics X - Proceedings of the Tenth Workshop on the Algorithmic Foundations of Robotics, WAFR 2012, MIT, Cambridge, Massachusetts, USA, June 13-15 2012 (Springer Tracts in Advanced Robotics, Vol. 86)*, Emilio Frazzoli, Tomás Lozano-Pérez, Nicholas Roy, and Daniela Rus (Eds.). Springer, 157–173.

Jingjin Yu and Steven M LaValle. 2013. Planning optimal paths for multiple robots on graphs. In *2013 IEEE International Conference on Robotics and Automation*. 3612–3617.

Neng-Fa Zhou, Håkan Kjellerstrand, and Jonathan Fruhman. 2015. *Constraint solving and planning with Picat*. Vol. 11. Springer.

Xiao Zhou, Syurei Tamura, and Takao Nishizeki. 2000. Finding Edge-Disjoint Paths in Partial k-Trees. *Algorithmica* 26, 1 (2000), 3–30. https://doi.org/10.1007/S004539910002

Received 30 May 2025; accepted 9 March 2026