The Thirty-Ninth AAAI Conference on Artificial Intelligence (AAAI-25)

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

Foivos Fioravantes[1], Dušan Knop[1], Jan Matyáš Křišťan[1], Nikolaos Melissinos[1], Michal Opler[1]
[1]Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague,
Prague, Czech Republic
{foivos.fioravantes, dusan.knop, kristja6, nikolaos.melissinos, michal.opler}@fit.cvut.cz

Abstract

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 CONSTRAINT 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.

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 CONSTRAINT (MAPFCC for short) problem. This requirement can arise, for example, from the need to constantly communicate with a human operator (Amigoni, Banfi, and Basilico 2017). Sometimes, only a periodic connection might be sufficient (Hollinger and Singh 2012). On the other hand, applications in a video 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 withing 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, Sankur, and Schwarzentruber 2023).

We refer the reader to the next section for the formal definitions.

Both MULTIAGENT PATH FINDING and MULTIAGENT PATH FINDING WITH COMMUNICATION CONSTRAINT 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, Kjellerstrand, and Fruhman 2015; Barták et al. 2017). A wide range of heuristics is commonly used, such as those based on lo-

---

cal search (WHCA*, ECBS), SAT solvers (e.g. (Surynek et al. 2022)), or reinforcement learning (e.g. (Gupta, Egorov, and Kochenderfer 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 CONSTRAINT 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, Miller, and Spirakis 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, Ganian, and Kanj (2023); Fioravantes et al. (2024a) 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 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 FPTand 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 FPTalgorithm 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, Ordyniak, and Ramanujan (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 1). 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 2). We stress here that many minor-closed graph classes have 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.

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. 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 $s_0(a)$. At specific times, the agents are allowed to move to a neighboring vertex 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 $s_i \colon A \to V$.

In this paper we consider the MULTIAGENT PATH FINDING WITH COMMUNICATION CONSTRAINT (MAPFCC for short) problem. In this generalization of the classical MULTIAGENT PATH FINDING problem, each agent has the capacity 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 also part of the input.

In order to define what is a feasible solution, 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 $dist_G(u, v) \leq d$. 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 $s_1, \ldots, s_m$ 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 $a \in A$, $i \in [m]$,
2. for all $i \in [m]$ 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 $i \in [m]$, and
4. for every agent $a \in A$, we have $s_m(a) = t(a)$.

Moreover, we do not allow agents to share any edge between any two turns. For ease of exposition, we will refer to the condition 4. above as the *communication constraint*. A feasible solution $s_1, \dots, s_m$ has *makespan* m. Our goal is to decide if there exists a feasible solution of makespan $m \leq \ell$.

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 $f(k) \cdot \mathrm{poly}(n)$, 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 tractable* (FPT). A problem admitting such an algorithm is said to belong in 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*. Allow us to define it properly.

A *tree-decomposition* of G is a pair $(\mathcal{T}, \beta)$, where $\mathcal{T}$ is a tree rooted at a node $r \in V(\mathcal{T})$, $\beta \colon V(\mathcal{T}) \to 2^V$ 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. It is known that computing a tree-decomposition of minimum width 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.

The Problem is Very Hard

In this section we will prove the following theorem.

Theorem 1. *The* Multiagent Path Finding with Communication Constraint *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 $G = (V, E)$, whose vertex set V is partitioned into the k independent sets $S_1, \dots, S_k$. The question is whether there exists a clique on k vertices as an induced subgraph of G. 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 (Fellows et al. 2009).

Starting from an input of the k-MCC problem, consisting of a graph H whose vertex set is partitioned into the sets $S_1, \dots, S_k$, 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 as an induced subgraph.

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 $S_i = \{v_1^i, \dots, v_n^i\}$; we build the V_i gadget as follows. We begin with n paths with $k-1$ vertices each, each one corresponding to a vertex of S_i. So, for each $p \in [n]$, we have the path $P_p^i = v_{p,1}^i v_{p,2}^i \cdots v_{p,i-1}^i v_{p,i+1}^i \cdots v_{p,k}^i$, which excludes the vertex $v_{p,i}^i$. We then add the new path $a_1^i \dots a_{i-1}^i a_{i+1}^i \dots a_k^i$ and, for each $p \in [n]$, we add the edge $a_j^i v_{p,j}^i$, for each $j \in [k] \setminus i$. We say that the vertices $v_{p,1}^i$ and $v_{p,k}^i$, 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). This finishes the construction of the V_i gadget, illustrated in Figure 1. Next, for each $l, m \in [k]$ with $l < m$, we build the $E_{l,m}$ gadget. This gadget consists in a forest of edges, each one corresponding to an edge between the vertices of S_l and S_m in H. That is, there exist vertices $u_p^{l,m}$ and $u_q^{m,l}$ in $E_{l,m}$ such that $u_p^{l,m} u_q^{m,l} \in E(E_{l,m})$ if and only if there exist vertices $v_p^l \in S_l$ and $v_q^m \in S_m$ such that $v_p^l v_q^m \in E(H)$. We say that $u_p^{l,m}$ and $u_q^{m,l}$, 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 $\frac{k(k-1)}{2}$ different such gadgets in total. This finishes the construction of the two gadgets we will need.

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 $a_k^i a_1^{i+1}$.

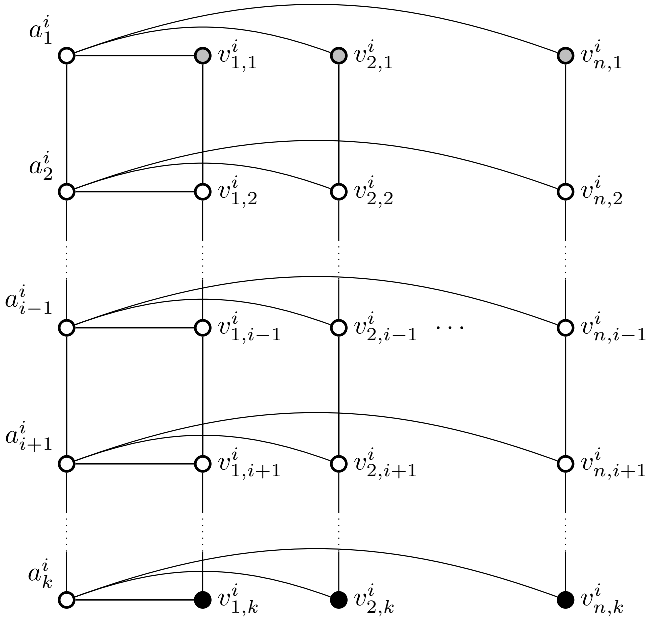

Figure 1: The V_i gadget, for any $i \in [k]$, used in the proof of Theorem 1. The color gray (black resp.) is used to represent the top (bottom resp.) vertices of the gadget.

Then, we connect the V_i gadgets with the $E_{l,m}$ gadgets as follows (illustrated in Figure 2). For every $l, m \in [k]$ with $l < m$ and $p, q \in [n]$, for every $u_p^{l,m} \in E_{l,m}$ and $u_q^{m,l} \in E_{l,m}$, we connect $u_p^{l,m}$ with $v_{p,m}^l$ and $u_q^{m,l}$ with $v_{q,l}^m$. 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 $\{t_j^i : i \in [k] \text{ and } j \in [i] \setminus j\}$, and we connect all the vertices of every $E_{l,m}$ gadget to all the vertices of Q. To finalise the construction of the instance I, we need to specify the set of agents, as well as the functions s_0 and t. For the set of agents, let $A = \{\alpha_j^i : i \in [k] \text{ and } j \in [k] \setminus i\}$. Then, for any $i \in [k]$ and $j \in [k] \setminus i$, let $s_0(\alpha_j^i) = a_j^i$ and $t(\alpha_j^i) = t_j^i$. 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 every agent is at distance exactly 3 from their ending position. Since in I we have that $\ell = 3$, it follows that any feasible solution of I will have makespan exactly 3 and will be such that $s_1(\alpha_j^i) \in P_p^i$, for some $p \in [n]$, $s_2(\alpha_j^i) \in E_{l,m}$, for some $l, m \in [k]$ with $l < m$, and $s_3(\alpha_j^i) = t_j^i$, for every $i \in [k]$ and $j \in [k] \setminus i$. Also, observe that in any feasible solution of I, we have that for every $i \in [k]$ there exists a unique $p \in [n]$ such that $s_1(\alpha_j^i) \in P_p^i$ for every $j \in [k] \setminus i$. Indeed, assume that s_1 is such that there exist $j < j' \in [k] \setminus i$ and $p < p' \in [n]$ with $v_1 = s_1(\alpha_j^i) \in P_p^i$ and $v_2 = s_1(\alpha_{j'}^i) \in P_{p'}^i$. Then $dist_G(v_1, v_2) \geq 2$. 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_j^i for some $j \in [k] \setminus i$; that is, there exists an $\alpha \in A$ such that $s_1(\alpha) = a_j^i$. This is a contradiction to the makespan of the solution.

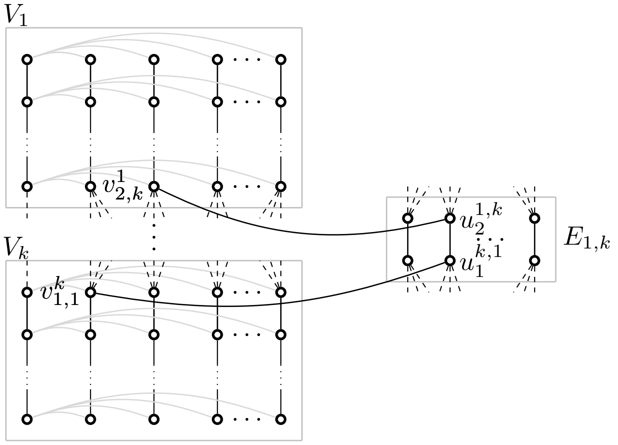

Figure 2: An example of the connection between the V_i and the $E_{l,m}$ gadgets used in the proof of Theorem 1. The edge $u_2^{1,k} u_1^{k,1}$ of $E_{1,k}$ represents the edge $v_2^1 v_1^k$ of H, where $v_2^1 \in S_1$ and $v_1^k \in S_k$. The dotted edges are used to represent the edges connecting the vertices between the corresponding gadgets.

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 $s_1(\alpha_j^i) \in P_p^i$ for every $j \in [k] \setminus i$; let us denote this p by $p(i)$. Consider now the vertices $v_{p(i)}^i$, 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 $v_{p(i)}^i v_{p(i')}^{i'} \notin E(H)$. Consider now the vertices $v_{p(i),p(i')}^i$ and $v_{p(i'),p(i)}^{i'}$ 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 $s_2(\alpha) = u_{p(i)}^{i,p(i')}$ and $s_2(\beta) = u_{p(i')}^{i',p(i)}$. This is a contradiction since, by its construction, the $E_{i,i'}$ gadget does not include these vertices since $v_{p(i)}^i v_{p(i')}^{i'} \notin E(H)$.

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 $\{v_{p(i)}^i : i \in [k]\}$ form a clique of H. For each $i \in [k]$ and $j \in [k] \setminus i$:

1. We set $s_1(\alpha_j^i) = v_{p(i),j}^i$. Clearly, $s_0(\alpha_j^i)$ is a neighbor of $s_1(\alpha_j^i)$. Moreover, since $v_{p(i),k}^i$ is connected to $v_{p(i+1),1}^{i+1}$ 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. $\square$

Efficient Algorithms

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

Few Agents and Short Communication

Theorem 2. *The* MULTIAGENT PATH FINDING WITH COMMUNICATION CONSTRAINT *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 CONSTRAINT. The algorithm is as follows. We build a 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 that is 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 CONSTRAINT if and only if the shortest directed path of H connecting u_s to u_t is of length at most ℓ. 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 $G[U]$ is connected and all agents are located in U. It is known (see Proposition 5.1. of (Guillemot and Marx 2014)) that given u, there exist at most $\Delta^{O(dk)}$ sets U of size dk where $u \in U$ $G[U]$ is connected. We can also enumerate them in time $\Delta^{O(dk)}$. Finally, we need to guess 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 $|V(H)| \leq \Delta^{O(dk)} kd^k n$. $\square$

Theorem 3. *The* MULTIAGENT PATH FINDING WITH COMMUNICATION CONSTRAINT *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 CONSTRAINT, 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 $3k$ and adapt appropriately the movements of the agents. Once this is done, it suffices to apply the algorithm provided in Theorem 2 to decide if I' is a yes-instance or not.

If $\Delta(T) \leq 3k$ then we are done. So, let $u \in V(T)$ such that $d_T(u) > 3k$. We may also assume that the makespan of I is optimal. For each $j \in [k]$, we define $P^-_{j,u}$ to be 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 $u = s_0(a_j)$ ($u = t(a_j)$ resp.), for some $j \in [k]$, then $P^-_{j,u} = \emptyset$ ($P^+_{j,u} = \emptyset$ resp.). Then, we define $P^-(u) = \{P^-_{1,u}, \dots, P^-_{k,u}\}$ and $P^+(u) = \{P^+_{1,u}, \dots, P^+_{k,u}\}$. Intuitively, the former (latter resp.) set contains all the paths of T that will be relevant for the agents to reach u (their targets resp.) from their initial positions (from u resp.). Finally, let $V_u = N(u) \cap (P^-(u) \cup P^+(u))$. That is, V_u contains u and its neighbors that are relevant in regards to the paths mentioned above. Observe that $|V_u| \leq 2k$. Let T_u be the connected 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 $v_1, \dots, v_p \in N(u) \setminus V_u$, for some $p > k$. We are now ready to describe the pruning that we perform: starting from T, delete all the vertices of T_u, except from u and k of its children $v_1, \dots, v_k$ in T_u; let T' be the resulting graph.

Claim 1. *If $I = \langle T, A, s_0, t, d, \ell \rangle$ is a yes-instance, then $I' = \langle T', A, s_0, t, d, \ell \rangle$ is also a yes-instance.*

Proof of the claim. Assume that $s = (s_1, \dots, s_\ell)$ is a feasible solution of I; we will construct $s' = s'_1, \dots, s'_\ell$ to be a feasible solution 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:

1. $\boldsymbol{s_i(a) \in T_u \setminus u}$ **and** $\boldsymbol{s_{i-1}(a) \in T_u \setminus u}$. Then $s'_i(a) = s'_{i-1}(a) = v_j$ for some $j \in [k]$.
2. $\boldsymbol{s_i(a) \in T_u \setminus u}$ **and** $\boldsymbol{s_{i-1}(a) \notin T_u \setminus u}$. 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 $s'_{i-1}(a) = s_{i-1}(a) = u$.
3. $\boldsymbol{s_i(a) \notin T_u \setminus u}$ **and** $\boldsymbol{s_{i-1}(a) \notin T_u \setminus u}$. 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.

Observe that the remaining case is symmetric to the second one detailed above.

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 $P^+_{j,u}$.

Finally, we will ensure that the connectivity constraint is preserved by s'. Let D and D' be the communication graphs of T and 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 $dist_T(s_l(b), s_l(c)) \leq d$ and $dist_{T'}(s'_l(b), s'_l(c)) > d$. 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 $s'_l(c) \notin T_u \setminus u$.** In this case, we have that $dist_{T'}(s'_l(b), s'_l(c)) \leq$ $dist_T(s_l(b), s_l(c)) - 1 \leq d - 1$.
- **$s'_l(b) \in T_u \setminus u$ 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, $dist(v_x, v_z) = 2 \geq d$.

Lastly, we deal with the particular case of $d = 1$. 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. 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'. ◇

As for the other direction of the equivalence, it holds trivially as T' is a subgraph of T. □

Tree-like Structures

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

Sketch of 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 $3\ell w$ 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 a standard use of the 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. Its vertex set is composed of sets $V_0, \ldots, V_\ell$ where V_i contains one copy of each vertex from V, i.e., $V_i = \{v_i \mid v \in V\}$. We refer to these sets as *layers* and we give all the vertices in V_i a vertex label $\mathsf{vertex}_\mathsf{i}$. 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_iv_i with an edge label inner.
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 $s_0(a)_0t(a)_\ell$ with en 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. 2024a) with the augmented graphs considered for edge-disjoint paths (Zhou, Tamura, and Nishizeki 2000; Ganian, Ordyniak, and Ramanujan 2021). Importantly, we observe that the existence of a solution is equivalent to an 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 $G[V_i]$.*

It is possible to show that Observation 1 can be translated into MSO_2 predicates. Unfortunately, it is not guaranteed that G_I must have small treewidth due to the edges connecting targets and destinations of 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 $O(\ell w)$.*

Proof of the claim. Let (T, β) be a tree decomposition of G of optimal width w. First, we consider a graph G'_I obtained from G_I by considering only the edges with labels copy, inner and cross. We define its tree decomposition (T, β') by replacing every occurrence of a vertex v in any bag with its $\ell + 1$ copies $v_0, \dots, v_\ell$. It is easy to see this is a valid tree decomposition of G'_I of width $O(\ell w)$.

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 guaranteed that $s_0(a)_0$ and $t(a)_\ell$ appear together in some bag, e.g., 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 $O(lw)$ as promised. ◇

Full algorithm. The algorithm first computes the treewidth w of G in time $2^{O(w^3)}n$ (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 Courcelle's theorem (Courcelle 1990) to evaluate φ on the obtained tree decomposition of G_I and outputs its answer. ◇

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. 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 labelled since 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 labelled 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 on planar graphs, and more generally on 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 bounded max-degree graphs.

Corollary 1. *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 2. *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.*

Conclusion

In this paper we initiated the study of the parameterized complexity of the MULTIAGENT PATH FINDING WITH COMMUNICATION CONSTRAINT problem. Our work 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 are used in practice. We consider all 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.

The full version of our work is available in (Fioravantes et al. 2024b).

References

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

Bannach, M.; and Berndt, S. 2019. Practical Access to Dynamic Programming on Tree Decompositions. *Algorithms*, 12(8): 172.

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *International Symposium on Combinatorial Search*, 19–27.

Barták, R.; Zhou, N.; Stern, R.; Boyarski, E.; and Surynek, P. 2017. Modeling and Solving the Multi-agent Pathfinding Problem in Picat. In *29th IEEE International Conference on Tools with Artificial Intelligence*, 959–966.

Bodlaender, H. L. 1996. A Linear-Time Algorithm for Finding Tree-Decompositions of Small Treewidth. *SIAM Journal on Computing*, 25(6): 1305–1317.

Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Betzalel, O.; Tolpin, D.; and Shimony, S. E. 2015. ICBS: The Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding. In *8th Annual Symposium on Combinatorial Search*, 223–225.

Calviac, I.; Sankur, O.; and Schwarzentruber, F. 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.

Courcelle, B. 1990. The Monadic Second-Order Logic of Graphs. I. Recognizable Sets of Finite Graphs. *Information and Computation*, 85(1): 12–75.

Cygan, M.; Fomin, F. V.; Kowalik, L.; Lokshtanov, D.; Marx, D.; Pilipczuk, M.; Pilipczuk, M.; and Saurabh, S. 2015. *Parameterized Algorithms*. Springer. ISBN 978-3-319-21274-6.

Downey, R. G.; and Fellows, M. R. 2012. *Parameterized complexity*. Springer Science & Business Media.

Eiben, E.; Ganian, R.; and Kanj, I. 2023. The Parameterized Complexity of Coordinated Motion Planning. In *39th International Symposium on Computational Geometry*, 28:1–28:16.

Eppstein, D. 2000. Diameter and Treewidth in Minor-Closed Graph Families. *Algorithmica*, 27(3): 275–291.

Fellows, M. R.; Hermelin, D.; Rosamond, F. A.; and Vialette, S. 2009. On the parameterized complexity of multiple-interval graph problems. *Theoretical Computer Science*, 410(1): 53–61.

Felner, A.; Stern, R.; Shimony, S. E.; Boyarski, E.; Goldenberg, M.; Sharon, G.; Sturtevant, N. R.; Wagner, G.; and Surynek, P. 2017. Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges. In *10th International Symposium on Combinatorial Search*, 29–37.

Fioravantes, F.; Knop, D.; Kristan, J. M.; Melissinos, N.; and Opler, M. 2024a. Exact Algorithms and Lowerbounds for Multiagent Path Finding: Power of Treelike Topology. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*, 17380–17388.

Fioravantes, F.; Knop, D.; Křišťan, J. M.; Melissinos, N.; and Opler, M. 2024b. Exact Algorithms for Multiagent Path Finding with Communication Constraints on Tree-Like Structures. arXiv:2412.08556.

Flum, J.; and Grohe, M. 2006. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer. ISBN 978-3-540-29952-3.

Ganian, R.; Ordyniak, S.; and Ramanujan, M. S. 2021. On Structural Parameterizations of the Edge Disjoint Paths Problem. *Algorithmica*, 83(6): 1605–1637.

Goldreich, O. 2011. *Finding the Shortest Move-Sequence in the Graph-Generalized 15-Puzzle Is NP-Hard*, 1–5. Berlin, Heidelberg: Springer. ISBN 978-3-642-22670-0.

Guillemot, S.; and Marx, D. 2014. Finding small patterns in permutations in linear time. In *Proceedings of the Twenty-Fifth Annual Symposium on Discrete Algorithms, SODA 2014*, 82–101.

Gupta, J. K.; Egorov, M.; and Kochenderfer, M. 2017. Cooperative Multi-agent Control Using Deep Reinforcement Learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops*, 66–83.

Hearn, R. A.; and Demaine, E. D. 2005. PSPACE-completeness of sliding-block puzzles and other problems through the nondeterministic constraint logic model of computation. *Theoretical Computer Science*, 343(1-2): 72–96.

Hecher, M. 2023. Advanced tools and methods for treewidth-based problem solving. *IT-Information Technology*, 65(1-2): 65–73.

Hollinger, G. A.; and Singh, S. 2012. Multirobot coordination with periodic connectivity: Theory and experiments. *IEEE Transactions on Robotics*, 28(4): 967–973.

Kloks, T. 1994. *Treewidth, Computations and Approximations*, volume 842 of *Lecture Notes in Computer Science*. Springer. ISBN 3-540-58356-4.

Korf, R. E. 1990. Real-time heuristic search. *Artificial intelligence*, 42(2-3): 189–211.

Korhonen, T.; and Lokshtanov, D. 2023. An Improved Parameterized Algorithm for Treewidth. In Saha, B.; and Servedio, R. A., eds., *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, 528–541. ACM.

Kornhauser, D.; Miller, G. L.; and Spirakis, P. G. 1984. Coordinating Pebble Motion on Graphs, the Diameter of Permutation Groups, and Applications. In *25th Annual Symposium on Foundations of Computer Science*, 241–250.

Langer, A. 2013. *Fast algorithms for decomposable graphs*. Ph.D. thesis, RWTH Aachen University.

Mouawad, A. E.; Nishimura, N.; Raman, V.; and Wrochna, M. 2014. Reconfiguration over Tree Decompositions. In *9th International Symposium on Parameterized and Exact Computation*, 246–257.

Niedermeier, R. 2006. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press. ISBN 9780198566076.

Sharon, G.; Stern, R.; Felner, A.; and Sturtevant, N. R. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial intelligence*, 219: 40–66.

Sharon, G.; Stern, R.; Goldenberg, M.; and Felner, A. 2013. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial intelligence*, 195: 470–495.

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

Snape, J.; Guy, S. J.; van den Berg, J.; Lin, M. C.; and Manocha, D. 2012. Reciprocal Collision Avoidance and Multi-Agent Navigation for Video Games. In *AAAI Workshop on Multiagent Pathfinding (MAPF@AAAI 2012)*.

Stern, R.; Sturtevant, N. R.; Felner, A.; Koenig, S.; Ma, H.; Walker, T. T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Barták, R.; and Boyarski, E. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *12th International Symposium on Combinatorial Search*, 151–158.

Surynek, P. 2010. An optimization variant of multi-robot path planning is intractable. In *AAAI Conference on Artificial Intelligence*, 1, 1261–1263.

Surynek, P. 2022. Problem Compilation for Multi-Agent Path Finding: a Survey. In *31st International Joint Conference on Artificial Intelligence*, 5615–5622. Survey Track.

Surynek, P.; Stern, R.; Boyarski, E.; and Felner, A. 2022. Migrating Techniques from Search-based Multi-Agent Path Finding Solvers to SAT-based Approach. *Journal of Artificial Intelligence Research*, 73: 553–618.

Tateo, D.; Banfi, J.; Riva, A.; Amigoni, F.; and Bonarini, A. 2018. Multiagent connected path planning: Pspace-completeness and how to deal with it. In *AAAI Conference on Artificial Intelligence*.

Wilson, R. M. 1974. Graph puzzles, homotopy, and the alternating group. *Journal of Combinatorial Theory, Series B*, 16: 86–96.

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

Zhou, N.-F.; Kjellerstrand, H.; and Fruhman, J. 2015. *Constraint solving and planning with Picat*, volume 11. Springer.

Zhou, X.; Tamura, S.; and Nishizeki, T. 2000. Finding Edge-Disjoint Paths in Partial *k*-Trees. *Algorithmica*, 26(1): 3–30.