Exact Algorithms and Lowerbounds for Multiagent Path Finding: Power of Treelike Topology

Foivos Fioravantes, Dušan Knop, Jan Matyáš Křišťan, Nikolaos Melissinos, Michal Opler

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

In the MULTIAGENT PATH FINDING problem, we focus on efficiently finding non-colliding paths for a set of k agents on a given graph G, where each agent seeks a path from its source vertex to a target. An important measure of the quality of the solution is the length of the proposed schedule ℓ, that is, the length of a longest path (including the waiting time). In this work, we propose a systematic study under the parameterized complexity framework. The hardness results we provide align with many heuristics used for this problem, whose running time could potentially be improved based on our Fixed-Parameter Tractability (FPT) results.

We show that MAPF is W[1]-hard with respect to k (even if k is combined with the maximum degree of the input graph). The problem remains NP-hard in planar graphs even if the maximum degree and the makespan ℓ are fixed constants. On the positive side, we show an FPT algorithm for $k + \ell$.

As we continue, the structure of G comes into play. We give an FPT algorithm for parameter k plus the diameter of the graph G. The MAPF problem is W[1]-hard for cliquewidth of G plus ℓ while it is FPT for treewidth of G plus ℓ.

Introduction

In this paper, we study the MULTIAGENT PATH FINDING (MAPF) problem. MAPF has many real-world applications, e.g., in warehouse management (Wurman, D'Andrea, and Mountz 2008; Li et al. 2021), airport towing (Morris et al. 2016), autonomous vehicles, robotics (Veloso et al. 2015), digital entertainment (Ma et al. 2017), and computer games (Snape et al. 2012). In MAPF the task is to find non-colliding paths for a set of k agents on a given graph G, where each agent seeks a path from its source vertex to a target.

This important problem has been, to the best of our knowledge, formally introduced about 20 years ago and attracted many researchers since then. Nowadays, there are numerous variants of the formal model for the problem; see (Stern et al. 2019). It is not surprising that the MAPF problem is in general NP-complete (Surynek 2010); the hardnes holds even in planar graphs (Yu 2016). Therefore, there are numerous heuristic approaches that allow us to efficiently obtain a useful solution for the given input; see, e.g., the survey (Stern 2019) for overview of such results. A multitude of techniques was used to tackle MAPF—A*-based algorithms (Hart, Nilsson, and Raphael 1968), SAT-based algorithms (Surynek et al. 2017), scheduling approach (Barták, Švancara, and Vlk 2018), SMT-solvers (Surynek 2019, 2020), to name just a few. Our work focuses on exact algorithms, that is, the aim is to return an optimal solution and the central question is on which kinds of inputs this can be done in an efficient way and where this is unlikely. While doing so, we initiate parameterised analysis of the MULTIAGENT PATH FINDING problem with the focus on natural and structural parameters. Note that the MAPF problem features two natural parameters—the number of agents k and the total length of the schedule ℓ (also called *makespan*). It is worth noting that two versions of MAPF are usually studied in the literature—one allows swapping two agents along an edge while this is prohibited in the other. Roughly speaking, the first version treats the input topology as a bidirected graph while the other as undirected. Most of our algorithmic results are independent of allowing or forbidding swaps; therefore, we reserve the name MULTIAGENT PATH FINDING for the version where we do not care about swaps. If the result holds only with swaps allowed, we use MULTIAGENT PATH FINDING SWAPS to refer to that specific version fo the problem while we use MULTIAGENT PATH FINDING NO SWAPS for the version where we explicitly do not allow swaps.

Our Contribution

It is well-known that MAPF can be reduced to a shortest-path problem in a graph known as the k-agent search space. Then, one can apply any algorithm to find a shortest path in this graph to find a solution to the original problem, e.g., using A*. The main downside of this approach is the size of the search space. We first show that it is unlikely to desing an efficient pruning algorithm for the k-agent search space:

Theorem 1. *The* MULTIAGENT PATH FINDING *problem is* W*[1]-hard parameterised by the number of agents* k *plus* $\Delta(G)$ *the maximum degree of the graph* G.

Indeed, the above observation yields the following.

Theorem 2. *The* MULTIAGENT PATH FINDING *problem is in* XP *parameterised by the number of agents* k.

We continue the study of classical complexity of MAPF with focus on the structure of the input graph G. Driven by some applications, one is interested in specific graph classes such as planar graphs or trees. Sadly, we show that the MAPF problem remains intractable in both these classes even if some other parts of the input are fixed constants.

Theorem 3. *The* MULTIAGENT PATH FINDING SWAPS *problem remains* NP*-complete even if the input graph G is planar, $\ell = 3$, and $\Delta(G) = 4$.*

It was known (Ma et al. 2016) that MULTIAGENT PATH FINDING is NP-complete when $\ell = 3$ (in general graphs). The NP-hardness for planar graphs (Yu 2016) was recently improved to constant degree (Eiben, Ganian, and Kanj 2023); there $\ell = 26$ and $\Delta(G) = 4$. However, Theorem 3 features all three of these properties. Furthermore, we believe that our reduction is simpler than the one of Eiben et al. It is not hard to see that instance with makespan 2 and swaps are allowed can be solved in polynomial time via a reduction to Hall's Marriage Theorem (which is used to find suitable middle point of all of the paths from sources to destinations). Thus, with the above theorem we get a tight dichotomy result in classical complexity.

Moreover we observe that, surprisingly, when swapping is not allowed, the problem becomes hard even for $\ell = 2$. Note that for $\ell = 1$ it is trivial.

Theorem 4. *The* MULTIAGENT PATH FINDING NO SWAPS *problem remains* NP*-complete even if the input graph G is planar, $\ell = 2$ and $\Delta(G) = 5$.*

The proof of Theorem 4 is achieved by slightly adjusting the gadgets used in the proof of Theorem 3.

Theorem 5. *The* MULTIAGENT PATH FINDING NO SWAPS *problem remains* NP*-complete even if G is a tree of maximum degree $\Delta(G) = 5$.*

Note that the MULTIAGENT PATH FINDING SWAPS was recently shown to be NP-complete for trees when the number of agents is equal to the number of vertices of that tree (Aichholzer et al. 2022). Our Theorem 5 differs from this work as, apart from treating the non-swapping version, it also tackles the problem for an arbitrary number of agents.

At this point, we see that none of the standard parameters—k or ℓ—is a good parameter alone. We complement this by showing that the combination of the two parameters results in fixed-parameter tractability.

Theorem 6. *The* MULTIAGENT PATH FINDING *problem is in* FPT *parameterised by the number of agents k plus the makespan ℓ.*

In the rest of the paper, we seek a good structural companion parameters to either of these. We begin with a very restrictive parameter—the vertex cover number that constitutes a rather simple starting point for our more general results. Here, we exploit the fact that many agents behave in a same way, i.e., they are almost anonymous. Furthermore, we can prune the input graph G, so that its size is bounded in terms of parameters (i.e., we provide a kernel).

Theorem 7. *The* MULTIAGENT PATH FINDING *problem is in* FPT *parameterised by the number of agents k plus the vertex cover number* $\mathrm{vc}(G)$.

It is known, that graphs of bounded vertex cover number have bounded diameter–the length of a longest shortest path in the graph (in a connected component). We strengthen the above result by proving that diameter is a strong companion to k. This yields tractability for many structural parameters including MIM-width as it implies bounded diameter. The proof uses the fact that if a graph has many vertices (unbounded in terms of k and the diameter) and small diameter, it must contain a vertex of large degree. We use such a vertex as a hub and prove that if we first route all agents to the neighborhood of that vertex and then to their respective destinations, we obtain routes with makespan $O(k \cdot d)$, where d is the diameter. The theorem then follows from Theorem 6.

Theorem 8. *The* MAPF *problem is in* FPT *parameterised by the number of agents k plus the diameter of G.*

Note that the vertex cover number $\mathrm{vc}(G)$ also bounds the number of agents that can move simultaneously. This can be generalized to a number of locally moving agents or, in other words, the size of separators in G. It is well-known that the size of vertex separators in a graph is related to a graph parameter treewidth. However, in view of our Theorem 5, there is little hope to conceive an efficient algorithm parameterized just by the treewidth of the input graph. To remedy this, we additionally parameterize by ℓ.

Theorem 9. *The* MULTIAGENT PATH FINDING *problem is in* FPT *parameterised by the treewidth* $\mathrm{tw}(G)$ *plus the makespan ℓ.*

Cliquewidth is a parameter further generalizing both vertex cover number and treewidth (Courcelle and Olariu 2000). However, the same additional parameterization as above leads here to intractability.

Theorem 10. *The* MULTIAGENT PATH FINDING *problem is in* W[1]*-hard parameterised by the cliquewidth* $\mathrm{cw}(G)$ *plus the makespan ℓ.*

Preliminaries

For integers m and n, we denote $[m : n]$ the set of all integers between m and n, that is, $[m : n] = \{m, m+1, \ldots, n\}$. We use $[n]$ as a shorthand for $[1 : n]$.

Formally, in the MULTIAGENT PATH FINDING problem we are given a graph $G = (V, E)$, a set of agents A, a positive integer ℓ, and two functions $s_0 \colon A \to V$ and $t \colon A \to V$ such that for any pair $a, b \in A$ where $a \neq b$, $s_0(a) \neq s_0(b)$ and $t(a) \neq t(b)$. Initially, each agent $a \in A$, is placed on the vertex $s_0(a)$. At specific times, called turns, 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 in the end of turn i (after the agents have moved) is given by a function $s_i \colon A \to V$.

We consider two versions of the problem; in MULTIAGENT PATH FINDING SWAPS (MAPFS) we allow *swaps*, *i.e.*, two agents to move through the same edge during the same turn, while in MULTIAGENT PATH FINDING NO SWAPS (MAPFNS) we do not allow swaps. In the first case, given $s_{i-1}(a)$ for every agent $a \in A$, *i.e.*, the positions of the agents at the turn $i-1$, the positions s_i are considered *feasible* if $s_i(a)$ is a neighbor of $s_{i-1}(a)$ in G, for every agent $a \in A$. In the second case, the positions s_i are feasible if, in addition to the previous, there is no pair of agents $a, b \in A$ such that $s_i(a) = s_{i-1}(b)$ and $s_i(b) = s_{i-1}(a)$.

We say that a sequence $s_1, \dots, s_\ell$ is a solution of $\langle G, A, s_0, t, \ell \rangle$ if s_i is considered feasible for all $i \in [\ell]$ and $s_\ell = t$. Also, a feasible solution $s_1, \dots, s_\ell$ has *makespan* ℓ. Our goal is to decide if there exists a solution of makespan ℓ.

Parametrized Complexity. Parametrized complexity is a computational paradigm that extends classical measures of time (and space) complexity, aiming to examine the computational complexity of problems with respect to a secondary measure—the parameter. Formally, a parameterized problem is a set of instances $(x, k) \in \Sigma^* \times \mathbb{N}$, where k is called the parameter of the instance. A parameterized problem is *fixed-parameter tractable* if it can be determined in $f(k) \cdot \mathrm{poly}(|x|)$ time for an arbitrary computable function $f\colon \mathbb{N} \to \mathbb{N}$. Such a problem then belongs to the class FPT. A parameterized problem is *slicewise polynomial* if it can be determined in $|x|^{f(k)}$ time for a computable function $f\colon \mathbb{N} \to \mathbb{N}$. Such a problem then belongs to the class XP. 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 2013) for a more comprehensive introduction to this topic.

Structural Parameters. Let $G = (V, E)$ be a graph. A set $U \subseteq V$ is a *vertex cover* if for every edge $e \in E$ it holds that $U \cap e \neq \emptyset$. The *vertex cover number* of G, denoted $\mathrm{vc}(G)$, is the minimum size of a vertex cover of G.

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 $\mathcal{T}$.

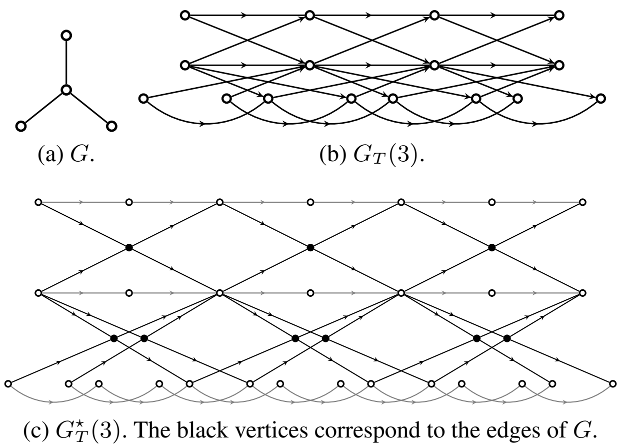
The *width* of a tree-decomposition $(\mathcal{T}, \beta)$ is $\max_{x \in V(\mathcal{T})} |\beta(x)| - 1$, and the treewidth $\mathrm{tw}(G)$ of a graph G is the minimum width of a tree-decomposition of G. It is known that computing a tree-decomposition of minimum width is fixed-parameter tractable 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).

Algorithms

For all the results presented in this section, we give a precise running time which was not given in the introduction.

Few Agents and Short Trips

Let $G = (V, E)$ be an undirected graph and ℓ a positive integer. We capture the movement of agents through G via a directed graph with vertices representing positions in both place and time. A *time-expanded graph of G with respect to ℓ*, denoted $G_T(\ell)$, is a directed graph with one copy of each vertex $v \in V$ for each time step $i \in [0 : \ell]$, i.e., its vertex set is $\{v_j \mid v \in V, j \in [0 : \ell]\}$. The set of vertices $\{v_j \mid v \in V\}$ for any fixed j, is called a *layer*. For every edge $\{u, v\} \in V$ and every $i \in [\ell]$, the graph $G_T(\ell)$ contains two arcs (u_{i-1}, v_i) and (v_{i-1}, u_i). Moreover for each vertex $v \in V$ and $i \in [\ell]$, the graph $G_T(\ell)$ contains an arc (v_{i-1}, v_i). Vertex-disjoint paths in $G_T(\ell)$ capture exactly the valid movements of agents when swaps are allowed.


(a) G. (b) $G_T(3)$.

(c) $G_T^\star(3)$. The black vertices correspond to the edges of G.

Figure 1: An example of a graph G, together with $G_T(3)$ and $G_T^\star(3)$.

We also introduce a modified version of this graph for the swap-free version of MULTIAGENT PATH FINDING. A *swap-free time-expanded graph of G with respect to ℓ*, denoted $G_T^\star(\ell)$, is a directed graph consisting of $2\ell + 1$ layers, that we again distinguish using subscripts. It contains a copy v_i of vertex $v \in V$ for every $i \in [0 : 2\ell]$. Moreover $G_T^\star(\ell)$ contains a vertex e_{2i-1} for every edge $e \in E$ and every $i \in [\ell]$. Similarly to $G_T(\ell)$, $G_T^\star(\ell)$ contains an arc (v_{i-1}, v_i) for every vertex $v \in V$ and every $i \in [2\ell]$. And finally for every edge $e = (u, v)$ in E and every $i \in [\ell]$, there are four arcs (u_{2i-2}, e_{2i-1}), (v_{2i-2}, e_{2i-1}), (e_{2i-1}, u_{2i}) and (e_{2i-1}, v_{2i}) in $G_T^\star(\ell)$. Again, it is straightforward to see that vertex-disjoint paths in $G_T^\star(\ell)$ capture exactly the valid movements of agents when swaps are disallowed.

Observation 1. *Let $\mathcal{I} = \langle G, A, s_0, t, \ell \rangle$ be an instance of* MAPF. *Then $\mathcal{I}$ is a yes-instance of* MAPFS *(*MAPFNS *resp.) if and only if there exists a set of directed pairwise vertex-disjoint paths in $G_T(\ell)$ ($G_T^\star(\ell)$ resp.) connecting all pairs $\{(s_0(a)_0, t(a)_\ell) \mid a \in A\}$ ($\{(s_0(a)_0, t(a)_{2\ell}) \mid a \in A\}$ resop.), where the second index is used to denote the corresponding layer of $G_T(\ell)$ ($G_T^\star(\ell)$ resp.).*

Theorem 11. *The* MULTIAGENT PATH FINDING *problem can be solved by an FPT-algorithm parameterised by the number of agents k plus the makespan ℓ in $O(2^{O(k \cdot \ell)} \cdot m \cdot \log n)$ time.*

Proof. We reduce to the BOUNDED VERTEX DIRECTED MULTI-TERMINAL DISJOINT PATHS (BVDMP) problem.

Its input consists of a directed graph G, two positive integers k, d, and k pairs of vertices $\{(s_i, t_i) \mid i \in [k]\}$ in G; and the task is to decide whether there is a set of k directed vertex-disjoint paths of length at most d connecting all pairs $\{(s_i, t_i) \mid i \in [k]\}$. The BVDMP problem can be solved by an FPT-algorithm parameterised by k plus d in $O(2^{O(k \cdot d)} \cdot m \cdot \log n)$ time (Golovach and Thilikos 2011).

Let $\langle G, A, s_0, t, \ell \rangle$ be an instance of MAPF. Due to Observation 1, it is sufficient to test whether there exists a set of k directed pairwise vertex-disjoint paths either in $G_T(\ell)$ or in $G^\star_T(\ell)$ depending on whether we allow swaps or not. Notice that such paths have length exactly ℓ in $G_T(\ell)$ and 2ℓ in $G^\star_T(\ell)$. Thus for MAPFS, it suffices to invoke the FPT algorithm for BVDMP on the graph $G_T(\ell)$ with the pairs of vertices $\{(s_0(a)_0, t(a)_\ell) \mid a \in A\}$ and setting $d = \ell$. While for MAPFNS, we run the same algorithm on the graph $G^\star_T(\ell)$ with the pairs of vertices $\{(s_0(a)_0, t(a)_{2\ell}) \mid a \in A\}$ and $d = 2\ell$. The resulting algorithms run in $O(2^{O(k \cdot \ell)} \cdot m \cdot \log n)$ time since both $G_T(\ell)$ and $G^\star_T(\ell)$ have $O(\ell \cdot m)$ edges. □

Tree-like Topology and Short Trips

Lemma 1. *For an undirected graph G and a non-negative integer p, the treewidth of both graphs $G_T(\ell)$ and $G^\star_T(\ell)$ is at most $O(\ell \cdot \mathrm{tw}(G))$.*

Proof. Let (T, β) be a tree decomposition of $G = (V, E)$ of optimal width. First, let us consider the graph $G_T(\ell)$. Let (T, β') be the tree decomposition obtained by replacing every occurrence of a vertex v in any bag with its $\ell + 1$ copies $v_0, \ldots, v_\ell$. Clearly, $|\beta'(x)| = (\ell+1) \cdot |\beta(x)|$ for every node x and it is easy to check that all the properties of tree decompositions hold.

Now, let us consider $G^\star_T(\ell)$. We first modify the tree decomposition (T, β) into (T', β') such that each edge e has a unique associated node x'_e in T' such that both endpoints of e lie in $\beta'(x'_e)$. That is easily achieved by first choosing an arbitrary such node x_e for each edge and then creating its new copy x'_e attached to x_e as a leaf. We construct a tree decomposition (T'', β'') of $G^\star_T(\ell)$ by the following modification of (T', β'). As before, we replace every occurrence of a vertex v in any bag with its $2\ell + 1$ copies $v_0, \ldots, v_{2\ell}$. Moreover, for each edge $e \in E$, we add the vertices $e_1, e_3, \ldots, e_{2\ell-1}$ to the bag $\beta''(x'_e)$. Observe that $|\beta''(x)| \le (2\ell + 1) \cdot |\beta(x)| + \ell$ for every node x in T''. It is again straightforward to verify the required properties of tree decompositions. □

Theorem 12. *The* MULTIAGENT PATH FINDING *problem can be solved by an FPT-algorithm parameterised by the treewidth w of G plus the makespan ℓ in $(\ell \cdot w)^{O(\ell \cdot w)} \cdot n$ time.*

Proof. Let $\langle G, A, s_0, t, \ell \rangle$ be an instance of MAPF. First, let us consider the variant MAPFS where swaps are allowed. Due to Observation 1, it suffices to check whether there exists a set of directed pairwise vertex-disjoint paths in $G_T(\ell)$ connecting all pairs $\{(s_0(a)_0, t(a)_\ell) \mid a \in A\}$. Moreover by Lemma 1, the treewidth of $G_T(\ell)$ is at most $O(\ell \cdot \mathrm{tw}(G))$. Finding vertex-disjoint paths in a directed graph $G_T(\ell)$ is in FPT parameterised by the treewidth $G_T(\ell)$ via a simple modification of the undirected case (Scheffler 1994) accounting for the orientation of the arcs. The algorithm runs in $(w')^{O(w')} \cdot n$ time, where w' is the treewidth of $G_T(\ell)$. Thus, we can solve the MULTIAGENT PATH FINDING problem in $(\ell \cdot w)^{O(\ell \cdot w)} \cdot n$ time, where w is the treewidth of G.

For the swap-free variant MAPFNS, the algorithm follows via the same argument simply by using the swap-free time-expanded graph $G^\star_T(\ell)$ instead of $G_T(\ell)$. Observe that $G^\star_T(\ell)$ has $O(\ell \cdot (n + m))$ vertices and $O(\ell \cdot (n + m))$ edges where n, m is the number of vertices and edges in G, respectively. However, we can bound m by $O(w \cdot n)$ since graphs of bounded tree-width are sparse (Kloks 1994) and thus the asymptotic bound on the runtime remains unchanged. □

Few Agents and Small Vertex Cover

Theorem 13. *The* MULTIAGENT PATH FINDING *problem, admits a kernel of size $O(2^{\mathrm{vc}} k)$, where* vc *is the size of a minimum vertex cover of the given graph and k the number of agents.*

Sketch of proof. Let $\langle G, A, s_0, t, \ell \rangle$ be an instance of MULTIAGENT PATHFINDING problem and $U \subseteq V(G)$ be a minimum vertex cover of G. For each subset $S \subset U$ we denote with V_S the subset of $V(G) \setminus U$ where $v \in V_S$ if and only if $N(v) = S$. Notice that any pair of vertices u, v that belong in the same set V_S for some $S \subset U$ are twins. For each S, we will select a set of vertices, which will be called representative set and denoted by U_S, as follows: if $|V_S| \le 3k$ then $U_S = V_S$, otherwise we select the U_S that satisfies the following properties:

- $U_S \subseteq V_S$,
- $U_S \supseteq V_S \cap \{s_0(a), t(a) \mid a \in A\}$ and
- $|U_S \setminus \{s_0(a), t(a) \mid a \in A\}| = k$.

The new instance is $\langle H, A, s_0, t, \ell \rangle$, where $H = G[U \cup \bigcup_{S \subseteq U} U_S]$ and $|V(H)| = |U \cup \bigcup_{S \subseteq U} U_S| = \mathrm{vc} + 2^{\mathrm{vc}} 3k$.

The equivalence of the two instances follows from the fact that we have included enough vertices in the sets U_S. Intuitively, the vertices of the independent set $V(G) \setminus U$ are partitioned into the sets V_S using their neighborhood S. Then we have included a sufficient number of vertices from each V_S, i.e., the sets U_S, for each $S \subseteq U$, so whenever an agent needs to move to a vertex $v \in V(G) \setminus U \cup \{s_0(a), t(a) \mid a \in A\}$ we can guarantee that there is always a twin of v, $u \in U_{N_G(v)}$, that can replace v in the new graph. ◇

Few Agents and Bounded Diameter Topology

Theorem 14. *The* MAPF *problem can be solved by an FPT-algorithm parameterised by the diameter d of G plus the number of agents k in $2^{O(k^2 \cdot d \cdot \log d)} \cdot m \cdot \log n$ time.*

The result is a consequence of the fact that in any sufficiently large graph G, there is always a feasible swap-free solution with makespan at most $O(d \cdot k)$.

Lemma 2. *Let $\langle G, A, s_0, t, \ell \rangle$ be an instance of* MAPF *such that for every $a \in A$, the vertex $t(a)$ is reachable from $s_0(a)$ and G has at least $(5 \cdot d \cdot k)^{d+1}$ vertices where d is the* *diameter of G and k is the number of agents. There exists a swap-free feasible solution of makespan $O(d \cdot k)$.*

Proof. Let us assume that G is connected as otherwise, the claim follows by considering each connected component separately. Any graph with maximum degree $\Delta > 2$ and diameter d has its number of vertices bounded by the Moore bound $1 + \Delta\frac{(\Delta-1)^d-1}{\Delta-2}$, see (Hoffman and Singleton 1960). Therefore, there is a vertex v of degree at least $5 \cdot d \cdot k$ in G.

Without loss of generality, let $A = [k]$ be the set of agents. For each agent $i \in A$, let P_i be an arbitrary shortest path between $s_0(i)$ and $t(i)$ and let H be the subgraph of G obtained as the union $\bigcup_{i\in[k]} P_i$. First, observe that H contains at most $k \cdot (d+1)$ vertices since each P_i is a shortest path in G. If H is disconnected, it consists of at most k connected components and we keep connecting disconnected components using shortest paths until we make it connected. Thereby, we obtain a connected graph on at most $2 \cdot k \cdot (d+1)$ vertices including the starting and target vertices of all agents.

Next, we construct a graph H' from H in the following way. If H does not contain the high-degree vertex v, we connect it by adding an arbitrary shortest path between v and H. In doing so, we increase the number of vertices by at most $d-1$. Moreover, we add to H' arbitrary k neighbors $w_1, \dots, w_k$ of v that are contained neither in H nor in the added shortest path between H and v. This is possible since there are at most $2 \cdot k \cdot (d+1) + d - 1 \leq 4 \cdot d \cdot k$ such vertices and v has at least $5 \cdot d \cdot k$ neighbors. In total, H' contains at most $O(d \cdot k)$ vertices.

Let T be an arbitrary spanning tree of H' and let $i_1, \dots, i_k$ be an ordering of the set of agents $[k]$ such that $\mathrm{dist}_T(s_0(i_j), v) \leq \mathrm{dist}_T(s_0(i_{j+1}), v)$ for every $j \in [k-1]$ where $\mathrm{dist}_T(x, y)$ is the distance between x and y in T. In other words, we order agents in increasing order with respect to the distance between their starting positions and v in T.

We construct a feasible solution with short makespan in the following way. We choose to describe the movements of all agents instead of tediously defining all the functions $s_1, s_2, \dots$. For $j \in [k]$, the agent i_j starts moving at time j and follows the shortest path from $s_0(i_j)$ to w_{i_j} in T without any further delay. We claim that there cannot be any conflicts. Assume for a contradiction that there exist $j, j' \in [k]$ such that the agents i_j and $i_{j'}$ collide at time p. Let us assume that $j < j'$. Observe that such a collision must occur only once the agents start moving and before they reach the vertex v because of the ordering by distances from v. In particular the agent i_j is at time p in distance $\mathrm{dist}_T(s_0(i_j), v) - p + j - 1$ from vertex v and the agent $i_{j'}$ is at distance $\mathrm{dist}_T(s_0(i_{j'}), v) - p + j' - 1$. However, we have $\mathrm{dist}_T(s_0(i_j), v) \leq \mathrm{dist}_T(s_0(i_{j'}), v)$ due to the way we defined the ordering and $j < j'$ by assumption. Therefore, the distance of the agent i_j is strictly smaller than that of agent $i_{j'}$ and we reach contradiction.

Once we stashed every agent $i \in [k]$ in the leaf w_i, we use the same strategy in reverse with respect to their target positions. In the first half of the process, it takes k turns before all the agents start moving and afterwards, they must all arrive at the leaves $w_1, \dots, w_k$ in at most $O(d \cdot k)$ turns since that is the total number of vertices in T. The same bound holds for moving the agents from $w_1, \dots, w_k$ to their target positions and thus, the total makespan of this solution is $O(d \cdot k)$. □

Proof of Theorem 14. Let n denote the number of vertices of G. If $n < (5 \cdot d \cdot k)^{d+1}$, we simply search the k-agent search space of size $(d \cdot k)^{O(kd)}$ using any of the standard graph searching algorithms (BFS, A*, etc.). Otherwise, we distinguish two cases. Either $\ell < C \cdot d \cdot k$ where the constant C is given by Lemma 2, and we invoke the FPT-algorithm of Theorem 6 on the given instance. Otherwise, we invoke the same FPT-algorithm on a modified instance with $\ell = C \cdot d \cdot k$ which is guaranteed to be equivalent by Lemma 2. □

Corollary 1. *Let $\mathcal{G}$ be a class of bounded treedepth, modularwidth, shrubdepth, or MIM-width. The* MAPF *problem is in FPT parameterised by the number of agents k if $G \in \mathcal{G}$.*

Proof. In each case, the diameter of G is bounded by some function of the respective parameter and thus, the result follows directly from Theorem 14. In fact, we show that in each case the respective parameter implies non-existence of long (induced) paths. A graph with treedepth w cannot contain path of length 2^w even as a subgraph, see (Nesetril and de Mendez 2012). Any graph class with bounded shrubdepth cannot contain arbitrarily long induced paths, see (Ganian et al. 2019). Modularwidth is monotone under taking induced subgraphs, and a path of n vertices has modularwidth n. Therefore, a graph with modularwidth k cannot contain an induced path of length k. Finally, the MIM-width of a graph G is defined as the size of the largest induced matching in G. Thus clearly, there cannot be an induced path of length more than $2w + 1$ in any graph with MIM-width w. □

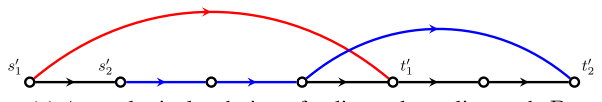
Hardness

Theorem 1. *The* MULTIAGENT PATH FINDING *problem is* W*[1]-hard parameterised by the number of agents k plus $\Delta(G)$ the maximum degree of the graph G.*

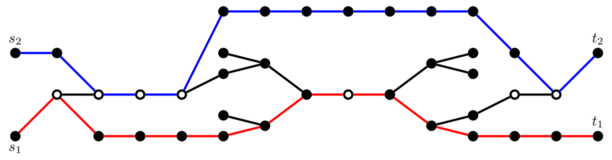
Sketch of proof. The reduction is from the k-DISJOINT SHORTEST PATHS (k-DSP) problem, which was shown to be W[1]-hard parameterised by k in (Lochet 2021), even when the graph given in the input is directed and acyclic. The input of this problem consists of a graph G and a set of pairs of vertices $\mathcal{P} = \{(s_i, t_i) : i \in [k]\}$; the question is whether there exists a set of (s_i, t_i)-paths hat are of minimum length and pairwise vertex-disjoint.

We construct an instance $\langle G, A, s_0, t, L\rangle$ of MULTIAGENT PATH FINDING which is a yes-instance if and only if the given instance $\langle D, \mathcal{P}'\rangle$ of the k-DSP problem is also a yes-instance. The value of L depends solely on the structure of $\langle D, \mathcal{P}'\rangle$ and is defined below.

To construct G, we first consider a topological ordering of the vertices of D. Through this ordering, we define *layers*, each one containing exactly one vertex of G', the underlying (undirected) graph of D. To simplify our sketch, we will refer to the vertices of D and G' using the same names. Then, for each (s', t') pair in $\mathcal{P}'$, we attach a leaf s (t resp.) to s' (t'

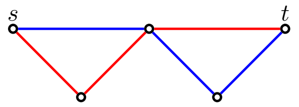


(a) A topological ordering of a directed acyclic graph D.

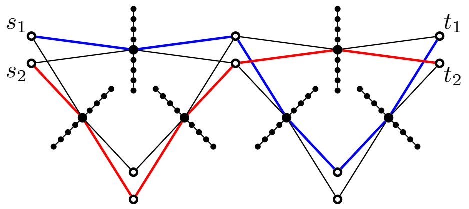


(b) The graph G constructed based on D.

Figure 2: An example for the construction used in the proof of Theorem 1. The color white denotes the original vertices, while the auxiliary vertices are colored black. The colors red and blue denote the two vertex-disjoint paths of D in subfigure (a), and the corresponding trajectories of G that the two agents will follow in subfigure (b). Vertices lying on the same vertical line in (b) belong in the same (sub-)layer.



(a) The graph G.



(b) The graph G'' constructed based on G.

Figure 3: An example for the construction used in the proof of Theorem 10 on a yes-instance $\langle G, s, t, 2, 3\rangle$ of BEUP. In subfigure (a), the colors red and blue distinguish two edge-disjoint paths of length 3. In subfigure (b), they denote the corresponding trajectories followed by the two path agents.

resp.), and define an agent a with $s_0(a) = s$ and $t(a) = t$. Then, we subdivide these newly added edges, so that each layer that lies before (after resp.) the layer of s' (t' resp.) receives exactly one additional vertex. We then replace the edges of the resulting graph that are incident to vertices of degree strictly greater than four, by a pair of binary trees (see Figure 2), and subdivide the newly added edges as before. In the resulting instance of MAPFNS, the starting positions of all the agents have the same distance from their ending positions, which is exactly equal to L. The main idea of the rest of the proof is that the agents will be able to reach their terminal positions in L turns if and only if they all move at each turn through k disjoint paths of G, which correspond to k disjoint paths of D. $\diamond$

Theorem 5. *The* MULTIAGENT PATH FINDING NO SWAPS *problem remains* NP*-complete even if G is a tree of maximum degree $\Delta(G) = 5$.*

Sketch of proof. We present a reduction from MAPFS, which was recently shown to be NP-hard on trees where the number of agents equals the order of the input graph (under the name of PARALLEL TOKEN SWAPPING in (Aichholzer et al. 2022)) to MAPFNS, where the input graph is a tree of maximum degree 5. Let $\langle T', A', s'_0, t', \ell'\rangle$ be an instance of MAPFS, where $|A'| = |V(T')|$ and T' is a tree. We construct an instance $\langle T, A, s_0, t, \ell\rangle$ of the MAPFNS problem and T is a tree of maximum degree 5, which is a yes-instance if and only if the starting instance is also yes-instance.

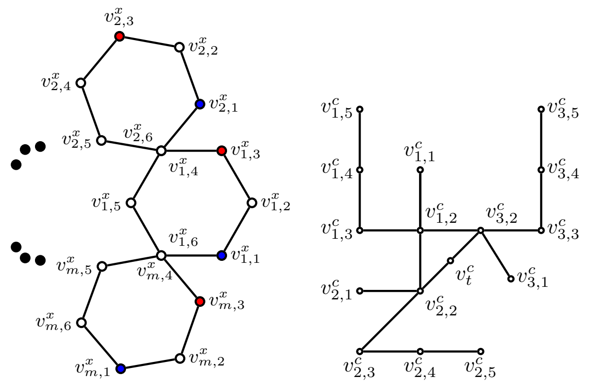
For the construction of T, we start with T', and for each non-leaf vertex v, we replace the set of edges that are leading to children of v by a binary tree, in a similar fashion as in the proof of Theorem 1. This results in a tree T of maximum degree 3. Let us say that the vertices of T that also belong to T' are the *original vertices* of T'. The agents of A are, for the moment, the same as those of A', with the same starting and ending positions. Then, we add two long *blocking paths* to each newly added vertex, resulting in T having maximum degree equal to 5. These paths are almost filled with *blocking agents*, which have starting and finishing positions on these paths. The purpose of these blocking agents is two-fold. First, in any solution of $\langle T, A, s_0, t, \ell\rangle$, these agents must move during each turn towards their ending positions, following a shortest path (along the blocking paths). Secondly, their initial placement allows us to have complete control over which turns these agents will block the original vertices of T'. Exploiting this control, we are then able to show the claimed equivalence. $\diamond$

Theorem 10. *The* MULTIAGENT PATH FINDING *problem is in* W*[1]-hard parameterised by the cliquewidth* $\mathrm{cw}(G)$ *plus the makespan ℓ.*

Sketch of proof. We reduce from the problem BOUNDED EDGE UNDIRECTED (s, t)-DISJOINT PATHS (BEUP): the input consists of a graph G with two distinct vertices s, t and two positive integers k, d; the question is whether there are k edge-disjoint (s, t)-paths of length at most d in G. The BEUP is W[1]-hard when parameterised by the treewidth of G for every fixed $d \geq 10$ (Golovach and Thilikos 2011).

Let $\langle G, s, t, k, d\rangle$ be an instance of BEUP where $G = (V, E)$. First, we subdivide each edge $e \in E$ with a new vertex v_e. Afterwards, we add a path P_e on $4d - 3$ vertices $v_e^1, v_e^2, \dots, v_e^{4d-3}$ such that its middle vertex v_e^{2d-1} is identified with v_e and all other vertices are disjoint from the rest of the graph. Finally, we replace each original vertex $v \in V$ with an independent set of k twins $v_1, \dots, v_k$. Let G'' be the constructed graph, see Figure 3 for an example. It can be shown that the cliquewidth of G'' is at most exponential in the treewidth of G.

The set of agents A consists of a set of *path agents* $A_0 = \{a_i \mid i \in [k]\}$ and a set of $2d - 2$ *edge agents* $B_e = \{b_e^i \mid i \in [2d - 2]\}$ for every edge $e \in E$. We set $s'_0(a_i) = s_i$, $t'(a_i) = t_i$ for every path agent $a_i \in A_0$. In other words, the starting and ending positions of path agents lie in the independent sets corresponding to the original vertices s and



(a) Variable gadget G^x, $m = m(x)$.

(b) Clause gadget H^C.

Figure 4: The two gadgets used in the proof of Theorem 3. In subfigure (a), the color red (blue resp.) is used on the vertices that may be incident to clause gadgets, in which x appear as a negative (positive resp.) literal.

t, respectively. For every $e \in E$ and every $i \in [2d-2]$, we set $s'_0(b^i_e) = v^i_e$ and $t'(b^i_e) = v^{2d+i-1}_e$.

Let us sketch the main idea of the equivalence between $\langle G, s, t, k, d\rangle$ and $\langle G'', A, s'_0, t', 2d\rangle$. For every original edge $e \in E$, each edge agent in B_e is exactly $2d-1$ steps away from its target and as a result, the edge agents of B_e cannot leave P_e nor change their order along it. It follows that throughout the whole process, at most one path agent can pass through the vertex v_e. On the other hand, we replaced every original vertex $v \in V$ with k twins and thereby, we simulate that any subset of path agents can occupy the vertex v at the same time. As a consequence, any set of k edge-disjoint paths of length at most d in G correspond precisely to the trajectories of path agents in G'' given by some feasible solution of makespan $2d$. $\diamond$

Theorem 3. *The* MULTIAGENT PATH FINDING SWAPS *problem remains* NP*-complete even if the input graph G is planar, $\ell = 3$, and $\Delta(G) = 4$.*

Sketch of proof. We present a reduction from the PLANAR 3-SAT problem which is known to be NP-complete (Garey and Johnson 1979). In that problem, a 3CNF formula ϕ on n literals and m clauses is given as an input. We construct an instance $\langle G, A, s_0, t, 3\rangle$ of MAPFS which is a yes-instance if and only if ϕ is satisfiable. For each variable x, let $m(x)$ be the number of times that x appears either as a positive or as a negative literal in ϕ.

We start the construction of G by defining a *variable* and a *clause* vertex for each variable and clause, respectively, that appears in ϕ. We add an edge between a variable and a clause vertex if the corresponding variable appears in the corresponding clause; let G' be the resulting graph. First, we replace each clause vertex c by H^c, which is a copy of the *clause gadget H* (illustrated in Figure 4(a)). Then, we replace each variable vertex x by G^x, a copy of the the *variable gadget G^x* (illustrated in Figure 4(b)). Note that all the edges of G' have been removed at this stage. So, for each edge xc in G', we add two new edges connecting vertices of G^x and H^c. In particular, to each $xc \in E(G')$, we assign an $i \in [m(x)]$ that hasn't already been assigned to a different appearance of x ϕ. Let x be in the j-th literal of C. If this literal is a positive one, then we add the edges $v^x_{i,1}v^c_{j,1}$ and $v^x_{i,1}v^c_{j,5}$. Otherwise, we add the edges $v^x_{i,3}v^c_{j,1}$ and $v^x_{i,3}v^c_{j,5}$. Let G be the resulting graph. Observe that, by carefully choosing and placing these edges, and since G' is planar, we can make sure that G is also planar.

We continue by defining the set of agents A and the functions s_0 and t. First, for each clause C, we create four *clause agents*, three of them denoted as a^c_i, $i \in [3]$, and the final as a^c. For $i \in [3]$, we set $s_0(a^c_i) = v^c_{i,1}$ and $t(a^c_i) = v^c_{i,4}$. Also, $s_0(a^c) = v^c_{1,2}$ and $t(a^c) = v^c_t$. Next, for each variable x, we create $m(x)$ *variable agents* a^x_i, $i \in [m(x)]$. For each $i \in [m(x)]$, we set $s_0(a^x_i) = v^x_{i,2}$ and $t(a^x_i) = v^x_{i,5}$. This completes our construction.

We are now ready to show that ϕ is a yes-instance of PLANAR 3-SAT if and only if $\langle G, A, s_0, t, 3\rangle$ is a yes-instance of MAPFS as well. First, assume that we have a satisfying assignment σ for ϕ. We define a feasible solution for $\langle G, A, s_0, t, 3\rangle$. The variable agents move through the variable gadgets in a counter-clockwise (clockwise resp.) fashion if $\sigma(x) = true$ ($\sigma(x) = false$ resp.). Next, for each clause C, we deal with the agents a^c_i, $i \in [3]$, and a^c. Since σ is a satisfying assignment, for each clause C there exists at least one literal x in C that satisfies it. The clause agent that corresponds to x moves through a variable gadget G^x. The other agents travel inside H^c.

The reverse direction follows directly from the fact that if the agents of H^c move only through edges of H^c, then any feasible solution will have a makespan of at least 4. In other words, at least one clause agent of each clause must move through the vertices of a variable gadget. We define an assignment of ϕ by taking into account the clockwise or counter-clockwise movement of the variable agents that satisfy the literal corresponding to the aforementioned clause agent. This is a satisfying assignment of ϕ. $\diamond$

Conclusion

In this paper we studied the parameterised complexity of the MULTIAGENT PATH FINDING problem. The main takeaway message is that the problem is rather intractable. Indeed, the problem remains hard even on trees. Hence, the treewidth of the input graph is highly unlikely to yield an efficient algorithm (under standard theoretical assumptions). This suggests that the current heuristic-oriented approach followed by the community is, in some sense, optimal. On the positive side, we showed that there are various combinations of parameters that lead to efficient algorithms, such as the number of agents plus the makespan. These positive results could potentially lead to improvements in practice, which should be the subject of a dedicated future study. The first step towards this direction is to check whether our algorithms can be utilised in tandem with some state-of-the-art heuristic algorithm, in order to obtain an improved result.

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

The full version of our work is available in (Fioravantes et al. 2023).

References

Aichholzer, O.; Demaine, E. D.; Korman, M.; Lubiw, A.; Lynch, J.; Masárová, Z.; Rudoy, M.; Williams, V. V.; and Wein, N. 2022. Hardness of Token Swapping on Trees. In Chechik, S.; Navarro, G.; Rotenberg, E.; and Herman, G., eds., *30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany*, volume 244 of *LIPIcs*, 3:1–3:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Barták, R.; Švancara, J.; and Vlk, M. 2018. A Scheduling-Based Approach to Multi-Agent Path Finding with Weighted and Capacitated Arcs. In André, E.; Koenig, S.; Dastani, M.; and Sukthankar, G., eds., *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, 748–756. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM.

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

Courcelle, B.; and Olariu, S. 2000. Upper bounds to the clique width of graphs. *Discret. Appl. Math.*, 101(1-3): 77–114.

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. 2013. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer. ISBN 978-1-4471-5558-4.

Eiben, E.; Ganian, R.; and Kanj, I. 2023. The Parameterized Complexity of Coordinated Motion Planning. In Chambers, E. W.; and Gudmundsson, J., eds., *39th International Symposium on Computational Geometry, SoCG 2023, June 12-15, 2023, Dallas, Texas, USA*, volume 258 of *LIPIcs*, 28:1–28:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Fioravantes, F.; Knop, D.; Křišťan, J. M.; Melissinos, N.; and Opler, M. 2023. Exact Algorithms and Lowerbounds for Multiagent Pathfinding: Power of Treelike Topology. arXiv:2312.09646.

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.; Hlinený, P.; Nesetril, J.; Obdrzálek, J.; and de Mendez, P. O. 2019. Shrub-depth: Capturing Height of Dense Graphs. *Log. Methods Comput. Sci.*, 15(1).

Garey, M. R.; and Johnson, D. S. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman. ISBN 0-7167-1044-7.

Golovach, P. A.; and Thilikos, D. M. 2011. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discret. Optim.*, 8(1): 72–86.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Trans. Syst. Sci. Cybern.*, 4(2): 100–107.

Hoffman, A. J.; and Singleton, R. R. 1960. On Moore Graphs with Diameters 2 and 3. *IBM J. Res. Dev.*, 4(5): 497–504.

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

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.

Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. K. S.; and Koenig, S. 2021. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, 11272–11281. AAAI Press.

Lochet, W. 2021. A Polynomial Time Algorithm for the k-Disjoint Shortest Paths Problem. In Marx, D., ed., *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms, SODA 2021, Virtual Conference, January 10 - 13, 2021*, 169–178. SIAM.

Ma, H.; Tovey, C. A.; Sharon, G.; Kumar, T. K. S.; and Koenig, S. 2016. Multi-Agent Path Finding with Payload Transfers and the Package-Exchange Robot-Routing Problem. In Schuurmans, D.; and Wellman, M. P., eds., *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, 3166–3173. AAAI Press.

Ma, H.; Yang, J.; Cohen, L.; Kumar, T. K. S.; and Koenig, S. 2017. Feasibility Study: Moving Non-Homogeneous Teams in Congested Video Game Environments. In Magerko, B.; and Rowe, J. P., eds., *Proceedings of the Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE-17), October 5-9, 2017, Snowbird, Little Cottonwood Canyon, Utah, USA*, 270–272. AAAI Press.

Morris, R.; Pasareanu, C. S.; Luckow, K. S.; Malik, W.; Ma, H.; Kumar, T. K. S.; and Koenig, S. 2016. Planning, Scheduling and Monitoring for Airport Surface Operations. In Magazzeni, D.; Sanner, S.; and Thiébaux, S., eds., *Planning for Hybrid Systems, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 13, 2016*, volume WS-16-12 of *AAAI Technical Report*. AAAI Press.

Nesetril, J.; and de Mendez, P. O. 2012. *Sparsity - Graphs, Structures, and Algorithms*, volume 28 of *Algorithms and combinatorics*. Springer. ISBN 978-3-642-27874-7.

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

Scheffler, P. 1994. *A Practical Linear Time Algorithm for Disjoint Paths in Graphs with Bounded Tree Width*. Fachbereich Mathematik: Preprint-Reihe Mathematik. Techn. Univ.

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 Felner, A.; Sturtevant, N. R.; Bekris, K. E.; and Stern, R., eds., *Multi-agent Pathfinding, Papers from the 2012 AAAI Workshop, MAPF@AAAI 2012, Toronto, Ontario, Canada, July 22, 2012*, volume WS-12-10 of *AAAI Technical Report*. AAAI Press.

Stern, R. 2019. *Multi-Agent Path Finding - An Overview*, volume 11866 of *Lecture Notes in Computer Science*, 96–115. Springer.

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 Surynek, P.; and Yeoh, W., eds., *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS 2019, Napa, California, 16-17 July 2019*, 151–159. AAAI Press.

Surynek, P. 2010. An Optimization Variant of Multi-Robot Path Planning Is Intractable. In Fox, M.; and Poole, D., eds., *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*. AAAI Press.

Surynek, P. 2019. Lazy Compilation of Variants of Multi-robot Path Planning with Satisfiability Modulo Theory (SMT) Approach. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2019, Macau, SAR, China, November 3-8, 2019*, 3282–3287. IEEE.

Surynek, P. 2020. Continuous Multi-agent Path Finding via Satisfiability Modulo Theories (SMT). In Rocha, A. P.; Steels, L.; and van den Herik, H. J., eds., *Agents and Artificial Intelligence, 12th International Conference, ICAART 2020, Valletta, Malta, February 22-24, 2020, Revised Selected Papers*, volume 12613 of *Lecture Notes in Computer Science*, 399–420. Springer.

Surynek, P.; Švancara, J.; Felner, A.; and Boyarski, E. 2017. Integration of Independence Detection into SAT-based Optimal Multi-Agent Path Finding - A Novel SAT-based Optimal MAPF Solver. In van den Herik, H. J.; Rocha, A. P.; and Filipe, J., eds., *Proceedings of the 9th International Conference on Agents and Artificial Intelligence, ICAART 2017, Volume 2, Porto, Portugal, February 24-26, 2017*, 85–95. SciTePress.

Veloso, M. M.; Biswas, J.; Coltin, B.; and Rosenthal, S. 2015. CoBots: Robust Symbiotic Autonomous Mobile Service Robots. In Yang, Q.; and Wooldridge, M. J., eds., *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 4423. AAAI Press.

Wurman, P. R.; D'Andrea, R.; and Mountz, M. 2008. Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. *AI Mag.*, 29(1): 9–20.

Yu, J. 2016. Intractability of Optimal Multirobot Path Planning on Planar Graphs. *IEEE Robotics Autom. Lett.*, 1(1): 33–40.