Solving Multiagent Path Finding on Highly Centralized Networks

Foivos Fioravantes[1], Dušan Knop[1], Jan Matyáš Křišťan[1], Nikolaos Melissinos[1], Michal Opler[1], Tung Anh Vu[2]

[1]Department of Theoretical Computer Science, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic

[2]Computer Science Institute, Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic

{foivos.fioravantes, dusan.knop, kristja6, nikolaos.melissinos, michal.opler}@fit.cvut.cz, tung@iuuk.mff.cuni.cz

Abstract

The MUTLIAGENT PATH FINDING (MAPF) problem consists of identifying the trajectories that a set of agents should follow inside a given network in order to reach their desired destinations as soon as possible, but without colliding with each other. We aim to minimize the maximum time any agent takes to reach their goal, ensuring optimal path length. In this work, we complement a recent thread of results that aim to systematically study the algorithmic behavior of this problem, through the parameterized complexity point of view.

First, we show that MAPF is NP-hard when the given network has a star-like topology (bounded vertex cover number) or is a tree with 11 leaves. Both of these results fill important gaps in our understanding of the tractability of this problem that were left untreated in the recent work of Fioravantes et al., Exact Algorithms and Lowerbounds for Multiagent Path Finding: Power of Treelike Topology, presented in AAAI'24. Nevertheless, our main contribution is an exact algorithm that scales well as the input grows (FPT) when the topology of the given network is highly centralized (bounded distance to clique). This parameter is significant as it mirrors real-world networks. In such environments, a bunch of central hubs (e.g., processing areas) are connected to only few peripheral nodes.

Introduction

Collision avoidance and optimal navigation are crucial in scenarios that require a group of agents to move through a network in an autonomous way. The real-world applications for such problems are well documented and range from virtual agents moving through a video game level (Snape et al. 2012), to actual robots transporting goods around a warehouse (Wurman, D'Andrea, and Mountz 2008; Li et al. 2021). Agents can be required to move through simpler topologies that span multiple "floors" (Veloso et al. 2015), or much more complicated topologies on a single "floor", e.g., in Airport surface management (Morris et al. 2016).

Many different models have been proposed to capture and solve the scenarios described above (Stern et al. 2019). This is due in part to the ubiquity of these scenarios, but also to the variety of slightly different requirements that each real-world application demands. For example, there are models where a set of positions must be reached, regardless of which agent reaches each one (Ali and Yakovlev 2024), while other models require each agent to reach a specific, predefined position. Many models consider the centralized setting (Ma et al. 2019; Sharon et al. 2015), while others consider the decentralized setting (Skrynnik et al. 2024).

The version we consider, usually appearing under the name of MULTIAGENT PATH FINDING (MAPF for short), consists of finding non-colliding paths for a set of agents that move through a given graph; each agent has to move from its source to its target vertex, both of which are predefined and unique, in an optimal way. That is, the agents move in such a way that the *length* of the schedule is minimized. To the best of our knowledge, this problem first started appearing more than 30 years ago (Hopcroft, Schwartz, and Sharir 1984; Reif 1979).

Unsurprisingly, the MAPF problem is computationally hard. In particular, it is NP-complete (Surynek 2010), and the hardness holds even in planar graphs (Yu 2016). Thus, the research community has mainly focused on heuristic algorithms that lead to efficiently solving the problem in practice. Some first positive results were based on the A^* algorithm (Hart, Nilsson, and Raphael 1968), while, more recently, many researchers have focused on exploiting highly efficient solvers; SAT-solvers were explored in (Surynek et al. 2017) and SMT-solvers in (Surynek 2019, 2020). There is also a plethora of greedy heuristic algorithms, such as the one presented in (Phan et al. 2024) which is based on *Large Neighborhood Search*. See (Stern et al. 2019) for a more comprehensive list of approaches of this flavor.

We follow a slightly different approach to attack this problem. Our goal is to design exact algorithms that are guaranteed to return an optimum solution. In view of the computational hardness of the MAPF problem, this is not possible in the general case. But what if we consider restricted instances? Can we identify characteristics, henceforth called *parameters*, whose exploitation leads to efficient algorithms? This is exactly the question that is in the core of the parameterized analysis that has recently been initiated for this problem (Eiben, Ganian, and Kanj 2023; Fioravantes et al. 2024a).

Our Contribution

The first two parameters that one would hope could be exploited are the number of agents k and the *makespan* ℓ, i.e.,

the total length of the schedule; these are exactly the two natural parameters of the MAPF problem. Unfortunately, the results of Fioravantes et al. (2024a) indicate that it is highly unlikely for either of these two parameters to be sufficient on their own. Thus, we turn towards structural parameters of the problem. Both (Eiben, Ganian, and Kanj 2023) and (Fioravantes et al. 2024a) provide a plethora of infeasibility results for many topologies which we could expect to come up in practical applications. Indeed, the MAPF problem remains NP-complete even when the input graph is a tree, or is planar and the makespan is at most 3.

Our first result addresses a gap in the tractability landscape that started being painted in (Fioravantes et al. 2024a) and it concerns graphs of bounded vertex cover number, a parameter whose exploitation usually leads to efficient algorithms (see, e.g., (Fellows et al. 2009)). Recall that the vertex cover of a graph is a set of vertices S such that all the edges of the graph are incident to at least one vertex of S. Note that the rest of the vertices of the graph form an independent set (there are no edges between them). This is the key observation, as it means that in any round of the schedule, no more than $|S|$ agents can move. So, intuitively, we could hope for an algorithm that would only focus on monitoring the agents that move through the vertices of S during every turn. Surprisingly, our first result shows that this approach will not work.

Theorem 1. MULTIAGENT PATH FINDING *remains* NP*-hard on trees with nine internal vertices and vertex cover number 7.*

We stress here that the above result is based on the tree having many leaves. In fact, if a tree has few internal vertices and leaves, the entire instance is bounded in size, and finding a solution cannot be intractable. Therefore, we ask the following question: What if the tree has only a few leaves? A generalization of this is captured by the parameter known as the max leaf number: the maximum number of leaves a spanning tree of a graph can have. Note that the max leaf number is a parameter that limits the structure of the input in a rather drastic way—the graph is essentially a subdivided tree with few edges (consequently, few internal vertices and leaves). Graphs admitting this structure appear quite often as transportation networks, in particular, underground trains (Eppstein 2015). Indeed, many problems are tractable with respect to the max leaf number (see, e.g., (Fellows et al. 2009)); a notable exception is the GRAPH MOTIF problem (Bonnet and Sikora 2017). Here, we obtain yet another intractability result for this parameter.

Theorem 2. MULTIAGENT PATH FINDING *remains* NP*-hard on trees with* 11 *leaves.*

Additionally, minor modifications to the previous theorem yield a hardness result in a more general setting involving semi-anonymous agents. The input of the COLORED MULTIAGENT PATH FINDING (COLORED MAPF) problem (Barták and Mestek 2021; Barták, Ivanová, and Svancara 2021) consists of a graph G, and multiple *types* of agents. All agents have their predefined starting positions, and each type of agents has a specific subset of the vertices of G which are the ending positions of that type; we are satisfied as long as every agent of a type ends up on *any* ending position of that type.

Theorem 3. COLORED MAPF *remains* NP*-hard even if there are only* 6 *groups of agents and* G *is a tree with* 11 *leaves.*

All the results we have so far seem to imply that having a sparse topology is not helpful. So, what about dense topologies? Our final result, which is the main algorithmic contribution of this paper, is an efficient algorithm that solves MAPFCC on graphs of bounded *distance to clique*. A graph G has distance to clique d if there is a set of d vertices S, whose removal renders the graph complete. Such topologies can be expected to appear in real-life applications where agents are required to move through a very dense network (the clique) which is connected to few docking stations (the vertices of S). We stress here that this is, to the best of our knowledge, the only positive result that exists that does not rely on the number of agents or makespan as parameters. In other words, the algorithm we construct remains useful even in instances where we have a large enough number of agents and/or long enough schedules that would make any previous exact algorithm infeasible in practice.

Theorem 4. MULTIAGENT PATH FINDING *is in FPT parameterized by the distance to clique.*

Preliminaries

We follow standard graph-theoretic notation (Diestel 2012). For $a, b \in \mathbb{N}$, let $[a, b] = \{c \in \mathbb{Z} \mid a \leq c \leq b\}$ and $[a] = [1, a]$. Let $G = (V, E)$ be a graph. For a subset of vertices $U \subseteq V$ and a vertex $u \in V$ we denote by $N_U(u)$ the set of vertices of U that are adjacent to u. By $N_U[u]$ we mean $N_U(u) \cup \{u\}$. As a short-hand, we write $N_G = N_{V(G)}$. When clear from the context, the subscripts will be omitted. Finally, for any $S \subseteq V$, let $G[S]$ be the subgraph of G that is induced by S, i.e., the graph that remains after deleting the vertices of $V \setminus S$ from G (along with their incident edges). For an m-dimensional integer vector $\vec{u} \in \mathbb{N}^m$, we write $\|\vec{u}\|_p = \left(\sum_{i=1}^{m} u_i^p\right)^{1/p}$, i.e. its ℓ_p-norm. By default, $\|\cdot\|$ denotes the ℓ_1-norm. For a function $f\colon A \to B$ and $C \subseteq A$ we write $f(C) = \{f(c) \mid c \in C\}$.

Formally, the input of the MULTIAGENT PATH FINDING (MAPF) problem consists of a graph $G = (V, E)$, a set of agents A, two functions $s_0\colon A \to V$ and $t\colon A \to V$ and a positive integer ℓ. 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, called *turns* or *rounds* (we will use both interchangeably), 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 an injective function $s_i\colon A \to V$.

We say that a schedule $s_1, \ldots, s_m$ is a *feasible solution* of an instance $\mathcal{I} = \langle G, A, s_0, t, \ell \rangle$ of MAPF if:

1. $s_i(a) \in N[s_{i-1}(a)]$ for all $a \in A$ and every $i \in [m]$,

2. $s_i(a) \neq s_i(b)$ for all $i \in [m]$ and $a \neq b \in A$, and
3. $s_m = t$.

A feasible solution $s_1, \ldots, s_m$ has *makespan* m. A feasible solution of *minimum* makespan will be called *optimum*. Our goal is to decide if there exists a feasible solution of makespan $m \leq \ell$.

It is worth mentioning here that there are two principal variations of MAPF. The first, more "generous" version, allows the agents to share edges if the wish to do so. For example, in this version, two agents a and b such that $s_i(a) = u, s_i(b) = v$, are allowed to move so that $s_{i+1}(a) = v, s_{i+1}(b) = u$, even if u and v are connected through a unique edge. The second, more "restrictive", version does not allow this behaviour. In this work we focus on the restrictive version.

Parameterized Complexity

Parameterized complexity is a field in algorithm design that takes into consideration additional measures to determine the time complexity. In some sense, it is about the multidimensional analysis of the time complexity of an algorithm, each dimension being dedicated to its own parameter. Formally, a parameterized problem is a set of instances $(x, k) \in \Sigma^* \times \mathbb{N}$; k is referred to as the *parameter*. The goal in this paradigm is to design *Fixed-Parameter Tractable* (FPT) algorithms, i.e., an algorithm that solves the problem $f(k)|x|^{O(1)}$ time for any arbitrary computable function $f\colon \mathbb{N} \to \mathbb{N}$. We say that a problem *is in FPT* if it admits an FPT algorithm. We refer the interested reader to classical monographs (Cygan et al. 2015; Downey and Fellows 2012) for a more comprehensive introduction to this topic.

Structural Parameters

There are three structural parameters that we consider in this work. Let $G = (V, E)$ be a graph.

First we have the *vertex cover number* of G. A set $S \subseteq V$ is called a *vertex cover* of G if for every edge $uv \in E$ we have that either $u \in S$ or $v \in S$ (or both). The vertex cover number of G is the size of a minimum vertex cover of G. This is a parameter that can be computed in FPT time, e.g., by (Chen, Kanj, and Xia 2006).

Next, we will consider the *distance to clique* of G. This is defined as the size of a minimum set $S \subseteq V$ such that $G[V \setminus S]$ is a clique (a graph where each vertex is adjacent to all other vertices of the graph). The distance to clique of G can be computed in FPT time by computing the vertex cover number of the *complement of* G. Recall that the complement of a graph G is the graph $G' = (V, E')$, where $uv \in E'$ if and only if $uv \notin E$.

Finally, the *max leaf number* of G is defined as the maximum number of leaves (i.e., vertices with only one neighbor) in a *spanning tree* of G. Recall that T is a spanning tree of G if the vertex set of T is the same as G and T contains no cycles. The max leaf number of a given graph can also be computed in FPT time (Fellows and Langston 1992).

Efficient Algorithm for Centralized Networks

In this section we prove the main positive result of this paper. First, we prove that the MULTIAGENT PATH FINDING problem can be solved in polynomial time on cliques; this will be necessary later on.

Theorem 4. MULTIAGENT PATH FINDING *is in FPT parameterized by the distance to clique.*

Sketch of proof. Let $\mathcal{I} = \langle G, A, s_0, t\rangle$ be an instance of MAPF. First consider a partition (M, Q) of $V(G)$ such that $|M| = dc$ and $G[Q]$ is a clique. We partition the agent set A into two sets A_1 and A_2, where A_1 includes exactly the agents whose starting or terminal position is in M, and $A_2 = A \setminus A_1$. The algorithm has 4 steps.

1. We prove that if $\mathcal{I}$ admits a feasible solution, then the makespan of any optimum solution of $\mathcal{I}$ is bounded by a computable function of dc.
2. We create a new instance $\mathcal{K} = \langle G', A', s_0', t'\rangle$ of MAPF such that:
   - G' is an induced subgraph of G with a number of vertices that is bounded by a computable function of dc.
   - $V(G') \supseteq M$.
   - $A' \supseteq A_1$ is selected appropriately.
   - If $\mathcal{I}$ admits a feasible solution $s_1, \ldots, s_m$ then $\mathcal{K}$ admits a feasible solution $s_1', \ldots, s_m'$ such that $|s_i'(A') \cap M| = |s_i(A) \cap M| \geq |A| - |Q|$.
3. We compute an optimum solution $s_1', \ldots, s_{m'}'$ of $\mathcal{K}$ such that $|s_i'(A') \cap M| \geq |A| - |Q|$ (if such a solution exists). This is done in FPT time parameterized by dc leveraging that $V(G')$ is bounded by a computable function of dc.
4. We extend the optimum solution $s_1', \ldots, s_{m'}'$ of $\mathcal{K}$ into an optimum solution $s_1, \ldots, s_{m'}$ of $\mathcal{I}$. This is done by creating consecutive matchings between the positions of the agents of $A \setminus A'$ during the round $i \in [m' - 1] \cup \{0\}$ and the empty positions in Q during the round $i + 1$, i.e., $Q \setminus s_{i+1}'(A')$.

It is easy to see that the above algorithm always returns an optimum solution of $\mathcal{I}$ if there exists any. Otherwise we can conclude that there is no feasible solution for $\mathcal{I}$.

Now, we give the idea behind each step.

Step 1. We define the PARTIALLY ANONYMOUS MULTIAGENT PATH FINDING (PAMAPF) problem, which is a generalization of MAPF. Let $\mathcal{J}$ be an instance of PAMAPF. The only thing that changes between $\mathcal{I}$ and $\mathcal{J}$ is the ending positions of the agents of A_2: in $\mathcal{J}$, these agents are allowed to finish on any vertex of Q that is the ending position of any agent in A_2 according to $\mathcal{I}$. Intuitively, this allows us to only keep track of the agents in A_1. Indeed, extending any feasible solution of $\mathcal{J}$ into a feasible solution of $\mathcal{I}$ is easily doable by solving MAPF in Q and only considering the agents of A_2. This can be done in at most two rounds.

Then it suffices to show that $\mathcal{J}$ admits a feasible solution with makespan bounded by a computable function of dc. To do that, we define different *snapshot types*, i.e., placements of the agents of A_1 in M; the snapshot types are defined

according to which agents of A_1 occupy which vertices and which vertices of M are not occupied at all throughout the schedule. The number of different snapshot types is at most $(|A_1|+2)^{|M|} = (2dc+2)^{dc}$. We then prove that in any optimum solution of $\mathcal{J}$, each snapshot type appears at most thrice. Indeed, if we assume that any snapshot type appears more than thrice, then we can replace the part of the schedule that lies between the first and last such appearances by a third occurrence of that same type.

Combining the arguments of the above two paragraphs, we can compute a feasible solution for $\mathcal{I}$ with makespan at most $3(2dc+2)^{dc}+2$.

Step 2. Hereafter we only consider feasible solutions of $\mathcal{I}$ of makespan at most $(3dc+2)^{dc}+2$. In any such feasible solution $s_1,\ldots,s_m$, there are at most $dc\big(3(2dc+2)^{dc}+2\big)$ agents $\alpha \in A$ such that $s_i(\alpha) \in M$. Also, $|s_i(A)\cap M| \geq |A|-|Q|$ since otherwise there would not be enough positions for the agents that are in Q during the round i.

We are ready to define $\mathcal{K}$. First, we partition the vertices of Q into $[2^{dc}]$ *vertex types* Q_τ, for $\tau \in [2^{dc}]$, based on their neighborhoods. We also partition the agents of A_2 into $[4^{dc}]$ *agent types* based on the vertex types of their starting and terminal positions. For any $\alpha \in A$, we set $\alpha \in A_{\tau,\sigma}$, for $(\tau,\sigma) \in [2^{dc}]\times[2^{dc}]$, if and only if $s_0(\alpha)\in Q_\tau$ and $t(\alpha)\in Q_\sigma$.

We begin by defining A', which is done inductively. We start with the set A^0 that includes A_1 and exactly $\min\{|A_{\tau,\sigma}|,(3dc+2)^{dc}+2\}$ agents from each agent type $A_{\tau,\sigma}$. Assume that we have constructed the set A^i for some $i\geq 0$. If there exists a vertex type $\phi\in[2^{dc}]$ such that $|Q_\phi|\leq 3|A^i|$ and $\bigcup_{\psi\in[2^{dc}]}(A_{\phi,\psi}\cup A_{\psi,\phi}) \not\subseteq A^i$, then we create a new set $A^{i+1}=A_i\cup\bigcup_{\psi\in[2^{dc}]}(A_{\phi,\psi}\cup A_{\psi,\phi})$. This process terminates because we cannot consider any vertex type Q_τ more than once. Observe also that $A^{i+1}\leq 7|A^i|$ as we must have $|Q_\phi|\leq 3|A^i|$ and $\sum_{\psi\in[2^{dc}]}(|A_{\phi,\psi}|+|A_{\psi,\phi}|)\leq 2|Q_\phi|$. Let A' be the set after finishing this process. Notice that $|A'|$ is bounded by $7^{2^{dc}}|A_0|\leq 7^{2^{dc}}\big(2^{dc}3((2dc+2)^{dc}+2)+|2dc|\big)$.

We now define G'. We start with a provisory set of vertices $U=M$. Then for each τ, we select $\min\{Q_\tau,3|A'|\}$ vertices from the vertex type Q_τ, and add them to U: first we select the starting and terminal vertices of the agents of A_1. The rest are selected arbitrarily. Notice that since τ and $|A'|$ are both bounded by computable functions of dc, the same holds for $|U|$. We set $G'=G[U]$.

Finally, we set s'_0 and t' to be the restrictions of s_0 and t to the set of agents A_1.

Then, we use the feasible solution $s_1,\ldots,s_m$ of $\mathcal{I}$ in order to prove that there exists a feasible solution $s'_1,\ldots,s'_m$ of $\mathcal{K}$ such that $|s'_i(A')\cap M|=|s_i(A)\cap M|\geq|A|-|Q|$. This is done by selecting an appropriate matching $N:A'\to A$ and using the positions of $s_i(N(\alpha))$ in order to define the positions $s'_i(\alpha)$ for all $\alpha\in A'$.

Step 3. For this we compute a graph H in which each vertex represents a placement of the agents of A' on the vertices of G'. Also, two vertices of H are adjacent if and only if we can move from the placement corresponding to one vertex to the one corresponding to the other in one round. Notice that the number of all different possible placements of the agents is bounded by $|V(G')|^{|A'|}$, which is also bounded by a computable function of dc. Therefore, the same holds for $|V(H)|$. Then, in order to compute an optimum solution of $\mathcal{K}$, it suffices to compute a shortest path between the vertices of H that represent s'_0 and t' respectively. If, additionally, we need to ensure that $|s'_i(A')\cap M|\geq|A|-|Q|$, it suffices to keep in H only the vertices that represent placements with this property. Since $V(H)$ is bounded by a computable function of dc, we can compute the wanted shortest path in FPT time, parameterized by dc.

Step 4. To extend the computed solution $s'_1,\ldots,s'_{m'}$ into an optimum solution of $\mathcal{I}$ we create an appropriate matching between the positions of the agents $A\setminus A'$ in round i and the vertices $Q\setminus s'_{i+1}(A')$. Notice that there are always enough positions for these agents in $Q\setminus s'_{i+1}(A')$ as we have guaranteed that $|s'_{i+1}(A')\cap M|\geq|A|-|Q|$. ◇

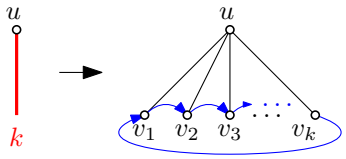

Figure 1: The operation of attaching a red edge of size k to vertex u in the proof of Theorem 1. Tails of blue arrows indicate the starting position of each agent and the heads indicate the agents' target position.

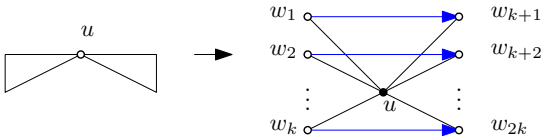

Figure 2: Attaching a bow tie of size k to a vertex u. The tails of the blue arrows indicate the starting vertex of an agent and the heads indicate the target vertex of an agent.

Trees With Few Internal Vertices or Leaves

This section is dedicated to the main infeasibility results we provide.

Theorem 1. MULTIAGENT PATH FINDING *remains* NP*-hard on trees with nine internal vertices and vertex cover number 7.*

Sketch of proof. We reduce from the 3-PARTITION problem, which is known to be NP-hard (Garey and Johnson 1975): as input we receive $3n$ integers $\vec{\beta}=(\beta_1,\ldots,\beta_{3n})$ where $\|\vec{\beta}\|=n\varphi$ for some $\varphi\in\mathbb{N}$. For a subset of indices $I\subseteq[3n]$ we denote $\beta(I)=\sum_{i\in I}\beta_i$. The goal is to partition $[3n]$ into n disjoint sets $\sigma_1,\ldots,\sigma_n$ of size 3 each so that $\beta(\sigma_i)=\varphi$ for every $i\in[n]$.

We start with the construction of a tree G and set of agents A (together with their starting and terminal positions) that does not yet depend on the instance of 3-PARTITION.

We construct G as follows. We start with a path of 5 vertices $u_1, \ldots, u_5$. Then we add four vertices u_6, u_7, u_8, u_9 together with the edges u_6u_2, u_8u_2, u_9u_2 and u_7u_4. These vertices will be the internal vertices of our tree. We proceed by creating two sets of vertices $\{x_1 \ldots, x_{4n-4}\}$ and $\{y_1, \ldots, y_{8n}\}$. Then, we add:

- x_iu_5, for all $i \in [2n-2]$,
- $x_{i+2n-2}u_6$, for all $i \in [2n-2]$,
- y_iu_8, for all $i \in [4n]$ and
- $y_{i+4n}u_9$, for all $i \in [4n]$.

The set A includes $6n-2$ agents $\alpha_1 \ldots \alpha_{6n-2}$. For all $i \in [4n]$, we set $s_0(\alpha_i) = y_i$ and $t(\alpha_i) = y_{i+4n}$, and for all $j \in [6n-2] \setminus [4n]$, we set $s_0(\alpha_j) = x_{j-4n}$ and $t(\alpha_j) = x_{j-2n-2}$.

Now, we will describe two operations. The first is the *attachment of a red edge of size k*, where $k \in \mathbb{N}$, to a vertex $u \in V(G)$ (see Figure 1). In this operation:

- we add k new vertices $v_1, \ldots, v_k$ to $V(G)$,
- we add all the edges v_iu, for $i \in [k]$, to $E(G)$,
- we create k new agents $a_1, \ldots, a_k$ such that $s_0(a_i) = v_i$ and $t(a_i) = v_{(i \mod k)+1}$ for each $i \in [k]$.

The second operation is the *attachment of a bow tie of size k to a vertex $u \in V(G)$* (see Figure 2): we add $2k$ vertices $w_1, w_2, \ldots, w_{2k}$ into $V(G)$, connect them all to u, add k new agents $a_1, \ldots, a_k$ into A, set $s_0(a_i) = w_1$ and $t(a_i) = w_{k+i}$ for every $i \in [k]$.

Next, we attach one bow tie of size $n\varphi+n+1$ to each one of the vertices u_3, u_5, u_6 and one bow tie of size $n\varphi-n-1$ to each one of the vertices u_8, u_9. Then, we attach to u_1 one red edge of size β_i for each $i \in [3n]$ (where β_i are the integers of the input of 3-PARTITION problem). Finally, we attach to u_7 one red edge of size φ, one red edge of size $\varphi+4$ and $n-2$ red edges of size $\varphi+2$. Let G be the resulting graph (see Figure 3) and A the final set of the agents. The construction of the instance $\mathcal{I} = \langle G, A, s_0, t, \ell \rangle$ of MAPF is completed by setting $\ell = n\varphi + 3n$.

On a high level, the reduction works as follows. First, the bow ties play the role of gates, forcing agents only move on the shortest path from their starting to their terminal positions. Then, the choice of ℓ is such that, according to any optimal solution of $\mathcal{I}$, the agents that start on $\{y_1, \ldots, y_{4n}\}$ and $\{x_1, \ldots, x_{2n-2}\}$ must constantly more towards their terminal vertices. This, essentially, sets a global clock according to which the vertices u_2 and u_4 are unoccupied only during specific rounds. Notice also that, in order for any agent that starts on a red edge to move towards their goal, they have to pass through either u_1 or u_7, according to which vertex their red edge is attached to. Also, in order to have enough empty vertices so the agents (that start on a red edge) can "rotate", one of them must occupy u_2 or u_4 when these are not occupied by any other agent (i.e. agents that start on some x_i or y_i). Moreover, we are able to prove that in any optimal solution of $\mathcal{I}$, once the first agent of any red edge from the βs (φs resp.) starts moving towards their terminal, then all the agents of that same red edge of the $\vec{\beta}$s (φs resp.) have to reach their terminal before another red edge of the same type can begin to be treated. This, combined with the global clock, ensures that the makespan of the schedule is exactly ℓ if and only if there are three red edges from the $\vec{\beta}$s that are dealt with every time that a red edge from the φs is completed, giving us the equivalence between the two problems. ◇

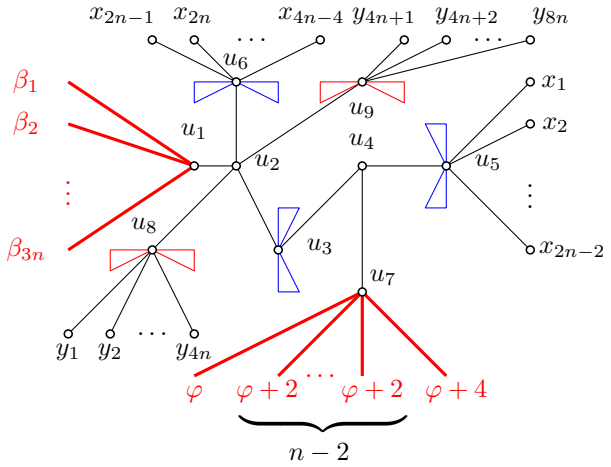

Figure 3: The graph G constructed in the proof of Theorem 1. Blue (red resp.) bow ties are of size $n\varphi + n + 1$ ($n\varphi - n - 1$ resp.).

Theorem 2. MULTIAGENT PATH FINDING *remains* NP*-hard on trees with* 11 *leaves.*

Sketch of proof. We reduce from the PANCAKE FLIPPING problem, which is known to be NP-hard (Bulteau, Fertin, and Rusu 2015): as input we receive a permutation $\pi = \pi_1, \ldots, \pi_n$ of the set $[n]$ together with a positive integer k. The question is whether π can be sorted using k prefix reversals, i.e., the operation of replacing a prefix of arbitrary length with its reversal. Note that we can assume that k is at most $\frac{18}{11}n$, otherwise it is trivially a yes-instance since every permutation of length n can be sorted using at most $\frac{18}{11}n$ prefix reversals (Chitturi et al. 2009).

On a high level, the reduction is as follows. Let $\langle \pi, k \rangle$ be an instance of PANCAKE FLIPPING where π is a permutation of length n and set $n^+ := n+2$, $L := 3n^+k$. We start with a description of a graph G' that forms the backbone of the final graph in the MAPF instance (see Figure 4. Observe that G is indeed a tree with exactly 11 leaves.). The graph G' consists of three paths, referred to as *A-path*, *B-path* and *C-path*, with their endpoints connected to a central vertex $v^\star$. The A-path is of length n, whereas both B- and C-paths are of length L. Let us denote the vertices of the A-path as $v_0^A, \ldots, v_n^A$, and the vertices of the B-path and C-path as $v_0^B, \ldots, v_L^B$ and $v_0^C, \ldots, v_L^C$ respectively, always starting from the neighbors of $v^\star$ outwards.

There are exactly n *primary agents* $a_1, \ldots, a_n$ that start on the vertices of the A-path in the order given by π, i.e., we set $s_0(a_i) = v_{\pi_i}^A$ for each $i \in [n]$. Their final destinations lie also on the A-path, this time in the increasing order of their indices, i.e., $t(a_i) = v_i^A$ for each $i \in [n]$. By the inclusion of

many additional *auxiliary agents*, situated on their respective auxiliary paths (see Figure 4), we will enforce that any optimum schedule has makespan L and moreover, it can be divided into exactly k rounds with each round consisting of three distinct phases — *push*, *reverse* and *pop*.

The four *auxiliary paths* of length $2L$ are as follows. Each path is connected with an edge to one of the vertices v_0^A, v_0^B or v_0^C. Namely, we add

1. a path with vertices $u_{-L}^A, \ldots, u_0^A, \ldots, u_L^A$ that is connected by an edge $u_{-1}^A v_0^A$,
2. a path with vertices $w_{-L}^A, \ldots, w_0^A, \ldots, w_L^A$ that is connected by an edge $v_0^A w_1^A$,
3. a path with vertices $u_{-L}^B, \ldots, u_0^B, \ldots, u_L^B$ that is connected by an edge $u_{-1}^B v_0^B$, and
4. a path with vertices $u_{-L}^C, \ldots, u_0^C, \ldots, u_L^C$ that is connected by an edge $u_{-1}^C v_0^C$.

Now let us define the auxiliary agents. First, there are L *auxiliary B-agents* $\{b_i^B \mid i \in [L]\}$ and L *auxiliary C-agents* $\{b_i^C \mid i \in [L]\}$. These groups of agents start in a single file on vertices $\{u_{-i}^B \mid i \in [L]\}$ and $\{u_{-i}^C \mid i \in [L]\}$ respectively. Their target destinations lie at distance exactly L from their starting positions, either on their starting auxiliary paths or on the B- or C-paths, respectively. We set $s_0(b_i^B) = u_{-i}^B$, $s_0(b_i^C) = u_{-i}^C$, and

$$t(b_i^B) = \begin{cases} v_{L-i}^B & \text{if } i \in [3\ell n^+ + 2n^+, 3\ell + 3n^+) \\ & \text{for some } \ell \in \mathbb{N}_0, \text{ and} \\ u_{L-i}^B & \text{otherwise.} \end{cases}$$

$$t(b_i^C) = \begin{cases} v_{L-i}^C & \text{if } i \in [3\ell n^+, 3\ell n^+ + n^+) \text{ for} \\ & \text{some } \ell \in \mathbb{N}_0, \text{ and} \\ u_{L-i}^C & \text{otherwise} \end{cases}$$

where $[a, b)$ denotes the set $\{a, a+1, \ldots, b-1\}$.

Finally, there are $L + 2n^+k$ *auxiliary A-agents* split into two sets. First, we have agents $\{b_{i,1}^A \mid i \in [L]\}$ with $s_0(\cdot)$ and $t(\cdot)$ defined as follows

$$s_0(b_{i,1}^A) = u_{-i}^A,$$

$$t(b_{i,1}^A) = \begin{cases} w_{L-i}^A & \text{if } i \in [3\ell n^+ + n^+, 3\ell + 2n^+) \\ & \text{for some } \ell \in \mathbb{N}_0, \text{ and} \\ u_{L-i}^A & \text{otherwise.} \end{cases}$$

Second, we add auxiliary A-agent $b_{i,2}^A$ for every $i \in [L] \cap \bigcup_{\ell \in \mathbb{N}_0} [3\ell n^+, 3\ell + n^+) \cup [3\ell + 2n^+, 3\ell + 3n^+)$ with the following starting and target positions

$$s_0(b_{i,2}^A) = w_{-i}^A, \quad t(b_{i,2}^A) = w_{L-i}^A.$$

Notice that every vertex u_{-i}^A for $i \in [L]$ is a starting position of some auxiliary A-agent, and every vertex w_{i-1}^A for $i \in [L]$ is a target position of some auxiliary A-agent. This concludes the description of the instance $\langle G, A, s_0, t, L \rangle$ of MAPF.

Each phase lasts exactly n^+ time steps. In the push phase, the vertex v_0^C is blocked and arbitrary prefix of primary agents with respect to the distance from $v^\star$ can move from the A-path to the B-path. In the reverse phase, the vertex v_0^A is blocked and thus, the A-path is separated from the B- and C-paths. Primary agents can move from the B-path to the C-path, effectively reversing their order with respect to the distance from $v^\star$. Finally in the pop phase, the vertex v_0^B is blocked, separating the B-path from the rest of the graph. All the primary agents located on the C-path now return to the A-path in reversed order. Moreover, auxiliary agents enforce that no primary agent can reside on the C-path during the push phase and similarly, that no primary agent can wait on the B-path during the pop phase. In this way, the primary agents can alter their order along the A-path in each round by performing a prefix reversal. Consequently, the primary agents can rearrange themselves into their final destinations on the A-path if and only if π can be sorted with k prefix reversals. ◇

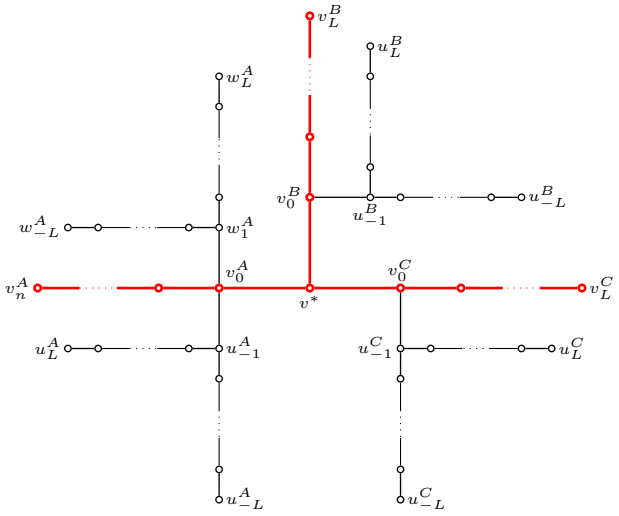

Figure 4: The graph G constructed in the proof of Theorem 2. The color red (black resp.) is used to indicate the main (auxiliary resp.) vertices and paths.

Additionally, a minimal modifications to the reduction above yield a hardness result in a more general setting where the agents are semi-anonymous. The input of a COLORED MULTIAGENT PATH FINDING (COLORED MAPF) problem consists of a graph G, positive integers k and ℓ and a group of agents A_i for each $i \in [k]$ with a prescribed starting positions S_i and target positions T_i. We are again looking for a feasible schedule of makespan at most ℓ but unlike in MAPF, we do not specify the target of each individual agent. We only require that the agents in A_i move from S_i to T_i for each $i \in [k]$.

Theorem 3. COLORED MAPF *remains* NP*-hard even if there are only* 6 *groups of agents and G is a tree with* 11 *leaves.*

Sketch of proof. We reduce from the BINARY STRING PREFIX REVERSAL DISTANCE problem known to be NP-hard (Hurkens et al. 2007): the input consists of two binary strings $\alpha = \alpha_1, \ldots, \alpha_n$ and $\beta = \beta_1, \ldots, \beta_n$ of length n,

and a positive integer k. The question is whether α can be transformed into β with k prefix reversals.

Let $\langle \alpha, \beta, k \rangle$ be an instance of BINARY STRING PREFIX REVERSAL DISTANCE. We refer to the reduction of Theorem 2. Observe that the constructed graph G and the starting and target positions of all auxiliary agents depend only on n and k. In particular, we do not make any modifications to the graph G. We shall also use the same set of agents, only split into six different groups.

The primary agents form the first two agent groups A_1 and A_2 where their starting and target positions correspond to the distribution of symbols 0 and 1 in the binary strings on input. Specifically, we set $S_1 = \{i \mid \alpha_i = 0\}$, $T_1 = \{i \mid \beta_i = 0\}$, $S_2 = \{i \mid \alpha_i = 1\}$ and $T_2 = \{i \mid \beta_i = 1\}$. The third and fourth group of agents consist of the auxiliary B-agents and auxiliary C-agents, respectively, with $S_3 = \{s_0(b_i^B) \mid i \in [L]\}$, $T_3 = \{t(b_i^B) \mid i \in [L]\}$ and S_4, T_4 defined analogously for the auxiliary C-agents. The final two groups of agents are formed by the two types of auxiliary A-agents. We set S_5 and T_5 to contain the starting and target position of each $b_{i,1}^A$ agent, respectively. And we define analogously S_6 and T_6 for the agents $b_{i,2}^A$.

Let us focus on the auxiliary B-agents, i.e., the third group A_3. Observe that within this group, there is a single possible pairing of positions in S_3 and T_3 such that each pair is at distance at most L and it exactly corresponds to the individual agents in the MAPF instance. The same holds for agent groups A_4, A_5 and A_6. Therefore, the movement of auxiliary agents in any arbitrary schedule of makespan L is predetermined and, in particular, enforces the same structure of the push, pull, pop phases described in the sketch of Theorem 2. We conclude that the primary agents can rearrange themselves exactly by a prefix reversal in each round. Therefore, $\langle G, (A_i, S_i, T_i)_{i=1}^6, L \rangle$ is a yes-instance of COLORED MAPF if and only if $\langle \alpha, \beta, k \rangle$ is a yes-instance of BINARY STRING PREFIX REVERSAL DISTANCE. $\diamond$

Conclusion

In this paper we continued down the path of studying the parameterized complexity of the MULTIAGENT PATH FINDING problem, an approach that has recently started gathering interest. Although the intractability of the problem had already been established even for restrictive cases, our hardness results are quite unexpected. Indeed, we stress a graph having bounded vertex cover or max leaf number has a very restricted structure, which leads to FPT algorithms the vast majority of times. Thus, our results in Theorems 1 and 2 are of great theoretical importance. Indeed, MAPF has the potential to be among the few candidates to give birth to reductions that show hardness when considering these parameters for a plethora of other problems. Moreover, these results provide further motivation towards a more heuristic approach. On the other hand, the FPT algorithm we presented has the potential to be of great practical use, depending on the specific characteristics of the topology on which the MAPF problem has to be solved, which can be the object of a dedicated study.

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. TAV was partially supported by Charles Univ. project UNCE 24/SCI/008 and partially by the project 22-22997S of GA ČR.

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

References

Ali, Z. A.; and Yakovlev, K. S. 2024. Improved Anonymous Multi-Agent Path Finding Algorithm. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*, 17291–17298.

Barták, R.; Ivanová, M.; and Svancara, J. 2021. From Classical to Colored Multi-Agent Path Finding. In *Fourteenth International Symposium on Combinatorial Search*, 150–152.

Barták, R.; and Mestek, J. 2021. OzoMorph: Demonstrating Colored Multi-Agent Path Finding on Real Robots. In *Thirty-Fifth AAAI Conference on Artificial Intelligence*, 15991–15993.

Bonnet, E.; and Sikora, F. 2017. The Graph Motif problem parameterized by the structure of the input graph. *Discrete Applied Mathematics*, 231(SI): 78–94.

Bulteau, L.; Fertin, G.; and Rusu, I. 2015. Pancake Flipping is hard. *Journal of Computer and System Sciences*, 81(8): 1556–1574.

Chen, J.; Kanj, I. A.; and Xia, G. 2006. Improved Parameterized Upper Bounds for Vertex Cover. In *Mathematical Foundations of Computer Science 2006, 31st International Symposium*, 238–249.

Chitturi, B.; Fahle, W.; Meng, Z.; Morales, L.; Jr., C. O. S.; Sudborough, I. H.; and Voit, W. 2009. An (18/11)n upper bound for sorting by prefix reversals. *Theoretical Computer Science*, 410(36): 3372–3390.

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.

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

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. 2015. Metric Dimension Parameterized by Max Leaf Number. *Journal of Graph Algorithms and Applications*, 19(1): 313–323.

Fellows, M.; Lokshtanov, D.; Misra, N.; Mnich, M.; Rosamond, F.; and Saurabh, S. 2009. The complexity ecology of parameters: An illustration using bounded max leaf number. *Theory of Computing Systems*, 45: 822–848.

Fellows, M. R.; and Langston, M. A. 1992. On Well-Partial-Order Theory and its Application to Combinatorial Problems of VLSI Design. *SIAM Journal on Discrete Mathematics*, 5(1): 117–126.

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.; Opler, M.; and Vu, T. A. 2024b. Solving Multiagent Path Finding on Highly Centralized Networks. arXiv:2412.09433.

Garey, M. R.; and Johnson, D. S. 1975. Complexity Results for Multiprocessor Scheduling under Resource Constraints. *SIAM Journal on Computing*, 4(4): 397–411.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Hopcroft, J.; Schwartz, J.; and Sharir, M. 1984. On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE- Hardness of the "Warehouseman's Problem". *The International Journal of Robotics Research*, 3(4): 76–88.

Hurkens, C. A. J.; van Iersel, L.; Keijsper, J.; Kelk, S.; Stougie, L.; and Tromp, J. 2007. Prefix Reversals on Binary and Ternary Strings. *SIAM Journal on Discrete Mathematics*, 21(3): 592–611.

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*, 11272–11281.

Ma, H.; Harabor, D.; Stuckey, P. J.; Li, J.; and Koenig, S. 2019. Searching with Consistent Prioritization for Multi-Agent Path Finding. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 7643–7650.

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 *Planning for Hybrid Systems, Papers from the 2016 AAAI Workshop*.

Phan, T.; Huang, T.; Dilkina, B.; and Koenig, S. 2024. Adaptive Anytime Multi-Agent Path Finding Using Bandit-Based Large Neighborhood Search. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*, 17514–17522.

Reif, J. H. 1979. Complexity of the Mover's Problem and Generalizations (Extended Abstract). In *20th Annual Symposium on Foundations of Computer Science*, 421–427.

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

Skrynnik, A.; Andreychuk, A.; Yakovlev, K. S.; and Panov, A. 2024. Decentralized Monte Carlo Tree Search for Partially Observable Multi-Agent Pathfinding. In *Thirty-Eighth AAAI Conference on Artificial Intelligence*, 17531–17540.

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. 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*, 3282–3287.

Surynek, P. 2020. Continuous Multi-agent Path Finding via Satisfiability Modulo Theories (SMT). In *12th International Conference on Agents and Artificial Intelligence*, 399–420.

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 *9th International Conference on Agents and Artificial Intelligence*, 85–95.

Veloso, M. M.; Biswas, J.; Coltin, B.; and Rosenthal, S. 2015. CoBots: Robust Symbiotic Autonomous Mobile Service Robots. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 4423.

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

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