ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230531

Intractability of Optimal Multi-Agent Pathfinding on Directed Graphs

Xing Tan [a;*] **and Pascal Bercher** [b;**]

[a]Department of Computer Science, Lakehead University, Canada
[b]School of Computing, The Australian National University

Abstract. In Multi-Agent Pathfinding (*MAPF*) problems, multiple agents move simultaneously to reach their individual destinations without colliding with each other. The computational complexity of the problem has been extensively studied for undirected graphs over the past decades. However, plan existence for Directed *MAPF* (*diMAPF*) was only recently studied and was shown to be in PSPACE as well as NP-hard. In this paper, we study the optimization versions (on makespan and on travel distance of agents) of *diMAPF* problems and show that they remain NP-hard even when various important non-trivial restrictions are imposed (e.g., when considering the problem on directed, acyclic, and planar graphs where the vertex-degrees are bounded). We have also provide membership results, thus presenting the first set of NP-completeness results for various optimal *diMAPF* variants.

1 Introduction

In a multi-agent pathfinding (MAPF) problem, multiple autonomous agents (e.g., robots in a warehouse) move simultaneously, searching and planning for paths to their respective destinations without colliding with each other [19, 16]. The problem is receiving increasing attention in both research and application in recent years [13, 11]. Relevance of MAPF can be found in several important problems/applications in the areas of robotics (e.g., in the coordination of autonomous drones or mobile robots in a factory), Operations Research (e.g., for optimizing transportation of goods and personnel in logistics and supply chain management), and AI planning [12, 2, 18, 15, 6, 14].

The problem of determining whether a multi-agent pathfinding (*MAPF*) problem has a solution can be solved efficiently in polynomial time for undirected graphs [7]. However, the optimization versions of the *MAPF* problem, including finding optimal solutions for makespan or travel distance of the agents, are NP-complete on general graphs and in various constrained settings, such as planar graphs or grids [17, 21, 20, 1, 5]. The case of MAPF on directed graphs (*diMAPF*) has only recently been studied and was proven to be NP-hard and in PSPACE [9]. However for restricted cases, such as directed acyclic graphs or strongly connected graphs, *diMAPF* becomes NP-complete [9, 10].

In this paper, we aim to go beyond the results of [9, 10] by examining the complexity of the optimization version of restricted *diMAPF* problems. Specifically, we focus on optimizing the makespan, where multiple agents move in time steps. Our study sheds light on this problem and provides insights into other optimization criteria, such as minimizing the maximal travel distance for any agent or minimizing the total travel distance for all agents. As shown in the paper, computational complexity of these optimal *diMAPF* problem, can be obtained based on our findings related to makespan.

Along this line of research we show that eventually, even if the underlying *diMAPF* problem is known to be solvable and the makespan is instantiated into its extreme value, it remains NP-hard to check the solvability of the corresponding optimization version for a given makespan. This complexity property is thus independent of the (NP-hard) reasoning about solvability of *diMAPF*, but is caused by the introduction of a makespan-bound to the problem. With a bounded makespan, *diMAPF* can be constrained on simple paths only for agents. Consequently, we are able to show NP-membership on top of its hardness, leading to the first set of NP-completeness results for optimal (i.e., makespan-bounded) *diMAPF*. These complexity results can be extended naturally to *diMAPF* bounded by maximal or overall travel distance as well.

To identify further sources of hardness, a variety of different restrictions to *diMAPF* aside from makespan, are considered. In fact, being able to extend the NP-completeness property from *diMAPF* on directed acyclic graphs [9] to problems where the graphs are in addition planar, marks an interesting breakthrough: Several other results follow almost immediately. For example, we are able to show readily NP-hardness of a highly restricted variant, the problem whose graph is not only planar, but also with vertex-degree bounded by three for all vertices. This result has a valuable implication on our future practice of algorithm design and development for real-world applications: We are now informed that under this practical scenario where agents are all in 2D space, on one-way roads with at most T-type junctions only, it is unrealistic to expect a poly-time algorithm for optimal solutions.

The remainder of this paper is organized as follows. Background knowledge including key concepts related to *MAPF* are defined in Section 2. Our complexity analysis is presented subsequently in Sections 3 and 4. Section 5 summarizes the obtained complexity results and concludes the paper.

2 Background

In this section, we formally define *MAPF* and *diMAPF* problems as well as the syntactic restrictions that we pose on them. We will provide additional definitions for restricting the makespan in later

* Email: xing.tan@lakeheadu.ca
** Email: pascal.bercher@anu.edu.au

sections once we start to investigate optimal (i.e., bounded) *diMAPF* problems.

Definitions follow the ones by Nebel in [9, 10]. Let $\mathcal{A}$ be a finite set of agents, and $G = \langle V, E\rangle$ a directed graph, where V is a finite set of vertices and $E \subseteq V \times V$ is a finite set of directed edges. An agent in $\mathcal{A}$ can move from $v_i \in V$ to $v_j \in V$ if $(v_i, v_j) \in E$ is an edge in the directed graph G (more constraints will be provided later). A state $\mathcal{S}$ defines a distribution of all agents from $\mathcal{A}$, in vertices from V. Formally, given $\mathcal{A}$ and $G = \langle V, E\rangle$ such that $|\mathcal{A}| \leq |V|$, the state $\mathcal{S}$ is defined to be an injective function $f : \mathcal{A} \to V$, so there cannot be any vertex collision (i.e., no vertex contains more than one agent).

Time is measured in steps. A step σ defines a step-wise movement of all agents, which changes a state $\mathcal{S}$ into its successor $\mathcal{S}^{succ}$. It is required that the movement of all agents in σ, between $\mathcal{S}$ and $\mathcal{S}^{succ}$, should be applicable ones, and the applicability of agent movements is defined by the principles of precondition and frame axioms in classical AI planning. That is, vertex v_j in $\mathcal{S}^{succ}$ contains an agent A if and only if:

1. Agent A is in v_j in $\mathcal{S}$ and remains there (and no other agent moves onto v_j); Or
2. Agent A is in some other vertex v_i in $\mathcal{S}$, and between these two successive states, A moves along (v_i, v_j) in V of G (no other agent moves onto v_j as well, and no other agent was on v_j in $\mathcal{S}$, unless it moves away from there).

Thus, movement onto v_j is allowed even if it was occupied before the move, as long as the respective agent moves away.

Let $\mathcal{S}_0 \equiv \mathcal{I}$ be the initial state, and $\mathcal{S}_n \equiv \mathcal{G}$ the final state where all the agents are injectively mapped into their respective goal-vertices. Let $\mathcal{S}_0$ be a state and $\Sigma \equiv \sigma_1, \ldots, \sigma_n$ a sequence of $|\Sigma| = n$ steps. Then, Σ applied to $\mathcal{S}_0$ is represented by $\vec{\mathcal{S}} \equiv (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_{n-1}, \mathcal{S}_n)$. That is, for all $1 \leq i \leq n$ step σ_i is applied to $\mathcal{S}_{i-1}$, resulting in a successor state $\mathcal{S}_i$. Given $\vec{\mathcal{S}}$, path p_i denotes the stepwise movement of agent A_i between states in $\vec{\mathcal{S}}$, initially from $\mathcal{S}_0 \equiv \mathcal{I}$ and eventually to $\mathcal{S}_n \equiv \mathcal{G}$ (i.e., $|p_i|$ is the travel distance of A_i)[1]. Set $\mathcal{P}_\Sigma$ contains all paths for all agents with a given Σ applied to $\mathcal{S}_0$, while path lengths are the travel distances for these agents. Thus p_i, the path of Agent A_i, is a member of $\mathcal{P}_\Sigma$. For the sake of brevity, from this point forward, we shall adopt the simplified notation $\mathcal{P}$ to represent $\mathcal{P}_\Sigma$, as long as it does not lead to any confusion or ambiguity. Note that the lengths of all paths in $\mathcal{P}$ might vary, but are upper-bounded by n, the total number of steps between states from $\mathcal{I}$ to $\mathcal{G}$. That is, an agent may not actually move between two successive states (e.g., in the example of Figure 1, the green agent A_g does not move between $\mathcal{S}_0$ and $\mathcal{S}_1$).

Definition 1 (*MAPF*) *A Multi-Agent Path Finding problem is a four-tuple $\langle G, \mathcal{A}, \mathcal{I}, \mathcal{G}\rangle$, where G is an undirected graph, $\mathcal{A}$ is a set of agents, $\mathcal{I}$ is the initial situation, and $\mathcal{G}$ is the goal situation. Is there a sequence $\vec{\mathcal{S}}$ that moves agents in $\mathcal{A}$ from $\mathcal{I}$ to $\mathcal{G}$?*

Definition 2 (*diMAPF*$^{\mathcal{R}}$) *A directed MAPF problem on a (possibly empty[2]) set of restrictions $\mathcal{R}$ is a MAPF problem $\langle G, \mathcal{A}, \mathcal{I}, \mathcal{G}\rangle$, where G is a directed graph that underlies the restrictions in $\mathcal{R}$. We consider the following restrictions:*

[1] Regarding the behavior of an agent after arriving at its goal vertex, in some literature (e.g., [16]) an agent will stay and will never move again – thus preventing any other agent from passing through it in future steps. We accept that any agent has the option to move out of its way, even if it is already on its goal vertex. To be explained next in our definitions, however, this issue is no longer relevant if the graph is a directed acyclic graph (DAG).

[2] If $\mathcal{R}$ is empty, we do not provide any superscript.

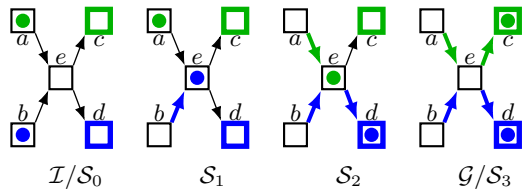

Figure 1: A *diMAPF* example. Initially at $\mathcal{I} \equiv \mathcal{S}_0$, the green agent A_g is at vertex a, and the blue agent A_b is at b. The goal for A_g is c, for A_b is d. In between $\mathcal{I}$ and $\mathcal{G}$, A_g waits at vertex a for A_b to pass vertex e first.

d* ≤ *3 *G is a DB3-digraph. Let "$dgr(i, j)$" denote a vertex in a digraph having in-degree i and out-degree j. A degree-bounded-by-3 (DB3)-digraph is a digraph such that for any vertex v in the graph, if $dgr(i, j)$ holds for v, then $i + j \leq 3$.*

dag *G is acyclic, i.e., a DAG[3].*

pl *G is planar. That is, it can be drawn on the plane in a way such that none of its edges intersect with each other.*

sc *G is strongly connected. A directed graph is strongly connected if every pair of vertices u and v should have a path in each direction between them.*

uc *G is unilaterally connected. That is, every pair of vertices u and v in G should have a path in at least one direction between them.*

wc *G is weakly connected. A directed graph is weakly connected if there is a path between every pair of vertices u and v in the underlying undirected graph.*

Note that, in terms of connectedness of a directed graph, from "sc", to "uc", to "wc", it is getting weaker. That is, a strongly connected graph is unilaterally connected, and a unilaterally connected graph must also be weakly connected. However such implications do not hold in the reversed direction.

Major results in [9, 10] are presented below, as we will discuss how they are related to our novel results.

Theorem 1 *[Theorem 1, Propositions 2 and 3, and Theorem 4 in [9], Theorem 18 and 19 in [10]] The problem diMAPF is*

1. *NP-hard, and in PSPACE;*
2. *NP-complete, if $\mathcal{G}$ is a directed acyclic graph (**dag**);*
3. *NP-complete, if $\mathcal{G}$ is a strongly connected digraph (**sc**)[4].*

3 On the Intractability of Severely Restricted Acyclic diMAPF Problems

Nebel [9] showed that unrestricted diMAPF problems are in PSPACE as well as NP-hard. He was furthermore able to show NP-completeness for a severely restricted case, namely for diMAPF problems on acyclic graphs. We refine this result by posing even further restrictions on the graph. *Specifically we show that the problem remains NP-hard even if the underlying graph is planar.*

We regard this a highly relevant result because real-world diMAPF problems often result from an actual environment that is being modeled, e.g., 2D landscapes, which are inherently planar. To prove NP-hardness on planar graphs, we use a highly restricted SAT problem.

[3] When a graph is a DAG, the question of whether an agent should be allowed to further move after its goal vertex, is no longer relevant, due to this DAG acyclicity property.

[4] This NP-completeness result is recently obtained in [10], through the proved validity of the Short Solution Hypothesis for strongly connected digraphs. That is, if an instance *diMAPF* in a given strongly connected digraph is solvable, a polynomial length solution exists for the instance.

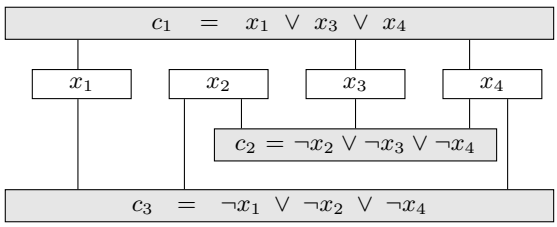

Figure 2: An example RPMS-3SAT instance.

A Boolean formula is a sentence of operations (conjunction, disjunction, or negation) of a set of n Boolean variables. The formula is in 3CNF if it is in the form of a conjunction of k clauses, and each clause is a disjunction of exactly three literals (a literal is a variable, or a negation of a variable). The problem 3SAT asks if a given 3CNF instance is satisfiable.

A 3CNF instance is *planar* if its corresponding bipartite incidence graph between variables and clauses is planar. In other words, there is an edge between variable x and clause c in the graph iff l, which is a literal corresponding to x, appears in c in the original formula. A 3CNF is *rectilinearly-planar* if its corresponding bipartite incidence graph is not only planar, but also possible to be embedded in a way where 1) all its variables can be arranged to form a straight line (e.g., from left to right with variables x_1 up to x_n), and 2) all its clauses are arranged into a rectilinear configuration. That is, each clause contains three "legs" and each leg connects the clause and one of the three variables corresponding to the three literals in the clause. Clauses are nested such that none of their legs will cross each other (thus the property of planarity is maintained). A 3CNF is *monotone* if its clauses are either all positive, or all negative.

Definition 3 *A Rectilinear-Planar Monotone Sided 3SAT (RPMS-3SAT) instance is a Boolean formula in the 3SAT format, rectilinearly-planar, and monotone. In addition, the clauses are sided: All positive/negative clauses are on the side above/below the variable line, respectively.*

Figure 2 is an example *RPMS-3SAT* instance, where the clause nodes are in grey color, and variable nodes are the white ones. The formula is

$$(x_1 \vee x_3 \vee x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_4)$$

The following theorem states the intractability of *RPMS-3SAT*, a property which is used to construct the NP-hardness part of the proof for Theorem 2.

Proposition 1 (Theorem 1 from [4]) *Rectilinearly-Planar Monotone Sided 3SAT (RPMS-3SAT) is NP-complete.*

It is stated in Proposition 2 by Nebel [9], that *diMAPF*^{dag} is NP-complete. We have the following result, which enforces in addition the planarity on DAGs in *diMAPF*. As will be demonstrated later, several theorems on optimal *diMAPF* problems can be derived almost immediately from this novel result.

Theorem 2 *diMAPF*^{dag,pl} *is NP-complete.*

Proof *Membership: diMAPF*^{dag,pl} is in NP, as it is a special case to the NP-complete *diMAPF*^{dag} problem.

Hardness: We perform a polynomial time transformation from *RPMS-3SAT* into the current problem. Given a *RPMS-3SAT* instance

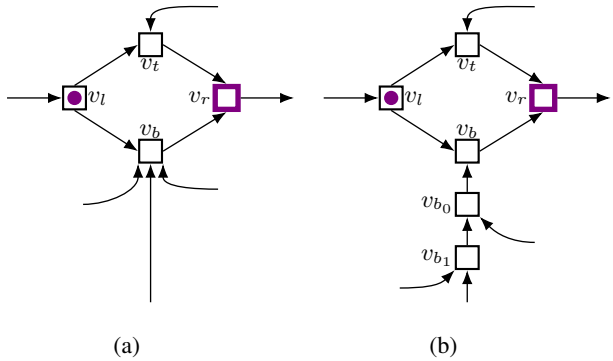

Figure 3: (a) A diamond component, constructed from a variable in the original *RPMS-3SAT* instance. For each variable in *RPMS-3SAT*, one and only one diamond component is generated. All these diamonds are to be connected into a chain (as shown in Figure 4). (b) A diamond variant to the one in (a). Two vertices (v_{b_0} and v_{b_1}) are added below v_b, and the variant now satisfies "d $\leq$ 3", a constraint that is studied in Theorem 11.

consisting of n variables and k clauses, we create a corresponding *diMAPF*^{dag,pl} instance $\langle G_{(dag,pl)}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$.

Each variable in *RPMS-3SAT* is converted into a "diamond" component (Figure 3.(a)) consisting of four vertices, the left, right, top, and bottom ones v_l, v_r, v_t and v_b, respectively. All edges are directed. Aside from the four edges connecting these four vertices (directions as indicated in Figure 3.(a)), there is exactly one edge entering the diamond into v_l and exactly one edge leaving the diamond from v_r. However there might be zero or multiple edges entering the diamond either from the top vertex v_t, or from the bottom one v_b (e.g., in Figure 3.(a), one edge enters from the top, and three edges enters from the bottom). There is a variable agent (called v-agent, in the color violet) initially based at v_l, having v_r as its goal vertex (highlighted in violet too).

All the diamonds are left-right connected into a chain (as illustrated in Figure 4). For each clause in *RPMS-3SAT*, we create a corresponding clause vertex, with three directed edges leaving it (corresponding to the three legs of the original clause). If the clause is positive, it will be on top of the diamond chain, otherwise it would be below the chain. According to the content of a given clause, the three leg-edges of the clause will enter their corresponding diamonds, respectively. For each clause vertex, there is a clause agent (called c-agent, in the color of cyan) initially in it.

Additionally there is a vertex (called B) entering the left-most vertex in the diamond chain. Initially there is an agent (called B-agent, highlighted in blue color, which will traverse the chain of diamonds, reflecting an assignment of Boolean values to all the variables in the original problem). There is also a linked chain of vertices connected to the right-most vertex in the diamond chain. These vertices are respectively the goal vertices for the c-agents, and at the end of the linked chain is B^g, the goal vertex for the B-agent.

The construction is complete, an example resulting *diMAPF*^{dag,pl} instance $\langle G_{(dag,pl)}, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$ is shown in Figure 4 (converted from Figure 2). Since the graph associated with the original *RPMS-3SAT* instance is planar, the resulting graph $G_{(dag,pl)}$ is planar too.

We have thus the following observations:

1. In order to arrive at its goal vertex, a c-agent has to traverse first at least one diamond component.
2. For any diamond component, it is never the case that a c-agent leaves it before the B-agent. Reason: If it happens, this c-agent

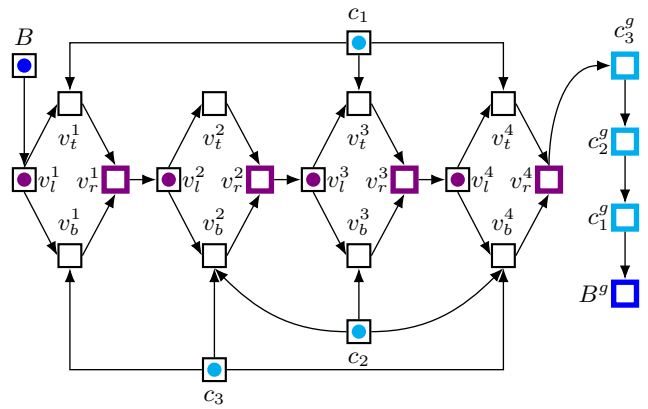

Figure 4: The resulting *diMAPF*^{dag,pl} instance $\langle G_{(dag,pl)}, \mathcal{A}, \mathcal{I}, \mathcal{G}\rangle$, converted from the RPMS-3SAT instance in Figure 2. Note in particular the graph $G_{(dag,pl)}$ is a DAG, and is planar.

will always be before the B-agent, and the problem can not be solved (regardless the problem is actually solvable or not).

3. To arrive at its goal vertex, the B-agent will have to traverse all the diamond components in a sequence. At each step, when it enters a diamond it will force the v-agent in the diamond to move.
4. There are exactly two alternative ways (through either the top vertex v_t, or the bottom vertex v_b) for a v-agent to arrive at its goal vertex v_r. On its way, the v-agent has to wait, to yield its way for the B-agent (and possibly c-agents too) to pass through the diamond component.
5. All along the path, the B-agent is left with no choice in terms of which vertex to explore. Its path is determined by all the v-agents. And when the B-agent is traversing a diamond, no c-clause will interfere. If a c-clause arrives at a diamond before (or even at one same step with) the B-agent, the c-agent will eventually block the B-agent from arriving its goal vertex B^g, which is located at the very end of the graph.

We are ready to prove a *RPMS-3SAT* instance is satisfiable iff the resulting *diMAPF*^{dag,pl} instance $\langle G_{(dag,pl)}, \mathcal{A}, \mathcal{I}, \mathcal{G}\rangle$ admits a solution.

($\Rightarrow$) : The satisfying truth-value assignment of variables in *RPMS-3SAT* corresponds to the path traversed by the B-agent through the diamonds (called the "blue-path"): If a variable v is assigned to be true/false, then the blue-path traverses v_t/v_b, respectively. Since all clauses are satisfied under this assignment, at least one leg of any clause enters this blue-path, thus the clause itself is able to arrive at its goal vertex. The B-agent leads, followed by the c-agents, which are lined up in exactly the order of $c_1 \dots c_k$, so all of them will be able to arrive at their goal vertices.

($\Leftarrow$) : If the resulting *diMAPF*^{dag,pl} admits a solution, the solution consists of a blue-path, and that blue-path corresponds to an satisfying truth-value assignment to the variables in the original *RPMS-3SAT* instance. Whichever diamond a c-agent chooses to go, it corresponds to the variable/litereal (assigned to be true) in the original clause. Thus the clause must also be true under this assignment. Hence the original *RPMS-3SAT* must be satisfiable. □

Corollary 3 *diMAPF*$^{dag,pl,d\leq 3}$ *is NP-complete.*

Proof *Membership: diMAPF*$^{dag,pl,\mathrm{d}\leq 3}$ is a special case of the NP-complete *diMAPF*^{dag,pl} problem, so membership follows directly.

Hardness: We can continue on the poly-time transformation performed in proving Theorem 2. Observe in particular that in the resulting graph (e.g., Figure 4), the degree constraint $\mathrm{d} \leq 3$ can possibly be violated only on those bottom/top vertices in the diamond components of the constructed graph. We can add up to k intermediated vertices before these bottom/top vertices such that the "leg" edges will enter these vertices first, instead of entering directly these bottom/top vertices. As a result, the $\mathrm{d} \leq 3$ constraint is satisfied, and the planarity property remains to be valid.

One such example is Figure 3. In Figure 3.(a), the $\mathrm{d} \leq 3$ constraint is violated at (and only at) the vertex v_b. However in Figure 3.(b), two vertices (v_{b_0} and v_{b_1}) are added below v_b, and the three edges initially enters v_b are now redirected accordingly, as illustrated in Figure 3.(b). □

Theorem 4 *diMAPF*^{dag,uc} *is NP-complete.*

Proof *Membership: diMAPF*^{dag,uc} is a special case of the NP-complete *diMAPF*^{dag} problem, so membership follows directly.

Hardness: We reduce from *diMAPF*^{dag} and create a unilaterally connected graph without having any side effects on solvability (or solution length).

We pair-wise check any two vertices, say u and v in G', to see whether u is reachable from v or vice versa (both can not be as this would contradict acyclicity). If one is reachable from the other, this pair is unilaterally connected already and there is nothing to do. If not, we introduce a new dummy vertex d and insert it via edges between u and v resulting into the path $u \to d \to v$. Doing so will not introduce a cycle as previously there does not exist a path between u and v. The resulting graph G'' remains a DAG but u and v are now unilaterally connected.

To ensure that solutions do not change, the new edges cannot be usable. For this, we simply introduce a new dummy agent a_d, place it initially on d and also define d as its goal. Since the graph is acyclic we know that if a_d moves, the agent will not be able to return, thus implying unsolvability. We do this for all pairs of vertices thus introducing at most $O(|V'|^2) = O(|V| \cdot |A|)^2$ new vertices, edges, and agents. □

4 On the Computational Complexity of Optimal diMAPF Problems (OdP)

In this section, we investigate the computational complexity of the optimization variant of *diMAPF* problems. Since we study decision problems, we model (as usual in complexity studies) the optimization criterion using an additional input, namely the makespan that is supposed to be optimized. Below, we first provide the additionally required definitions for this optimization criterion. Then, we present our results in three further sections. First, in Section 4.1, we present our results for the problem, where we pose restrictions on the graph only but no restrictions on the makespan. Then, in Section 4.2, we analyze how these results can be refined by further putting restrictions on the makespan. Finally, restrictions on the length of paths are studied in Section 4.3.

In terms of optimal *diMAPF* problems, we investigate various restricted variants of *diMAPF*, with three different optimization criteria considered.

Definition 4 (*diMAPF*$^{\mathcal{R}}_{\langle O,\boldsymbol{B}\rangle}$) *diMAPF*$^{\mathcal{R}}_{\langle O,B\rangle}$ *is a diMAPF problem subject to restrictions* $\mathcal{R}$ *and an optimization criterion of type O, which is bounded by* $B \in \mathbb{Z}^+$.

To decide diMAPF$^{\mathcal{R}}_{\langle O,B\rangle}$ *with* $O \in \{ms, \max, tot\}$ *means to answer: Is there a* $\vec{\mathcal{S}}$*, such that*

- *If $O = ms$, then:* $|\Sigma| \leq B$?
- *If $O =$* max*, then:* $\max_{p \in \mathcal{P}}(|p| \leq B)$?
- *If $O = tot$, then:* $\left(\sum_{p \in \mathcal{P}} |p|\right) \leq B$?

The set $\mathcal{R}$ now may also include the restriction ***sol*** *(solution provided), which means that the problem has an additional input, namely a solution to the given* unbounded *problem, i.e., it may not satisfy the given optimization criterion.*

Note that in Figure 1, $|\Sigma| = 3$ thus from the definition *ms* the makespan is 3. Each one of the two agents traverses two edges, thus *max* the maximal individual distance is 2. $\sum_{p \in \mathcal{P}} |p| = 2 + 2 = 4$, i.e., *tot* the total distance with two agents is 4.

Below we discuss briefly the relationships among these three optimization criteria in the general sense:

- $max \leq ms$. An agent might pause between two consecutive steps, and even if it non-stop moves for all steps, the best that can be achieved is $max = ms$.
- $max \leq tot$, which is obvious.
- For *ms* and *tot*, either one can be greater: Intuitively, in a scenario of $tot \leq ms$, a system might freeze itself between some steps leading to an arbitrary-long makespan, while the actual distance all agents travelled is relatively small. Alternatively, when in some another system where all the agents are routed effectively, it however can be the case $ms \leq tot$. A simple example is already included in Figure 1, where $ms = 3$ and $tot = 4$.

4.1 OdP with Graph Restrictions

We start with our results that (almost) directly follow from recent work by [9].

Corollary 5 *$diMAPF_{\langle ms,B \rangle}$ is NP-hard.*

Proof The proof is straightforward, given that we already know the problem *diMAPF* is NP-hard [9]. If *diMAPF* admits a solution, we can safely assume the solution does not repeat any state, otherwise we can simply remove all the states between any two repeated ones. Hence *diMAPF* is feasible iff it admits a sequence with length at most exponentially long, i.e., it has an upper bound of $|V|^{|\mathcal{A}|}$. We can use this bound as value of B, which is polynomial in space by encoding this exponential number logarithmically. □

In addition to NP-hardness and PSPACE membership of the general case, Nebel [9] also provides an NP-completeness proof, namely for acyclic graphs. This result can be directly transferred to our bounded case as well:

Corollary 6 *$diMAPF^{dag,uc}_{\langle ms,B \rangle}$ is NP-complete.*

Proof *Membership:* Due to acyclicity all paths for all agents can be limited by a polynomial, or more precisely by $|V|$. This means that the maximal makespan is bounded by $|V| \cdot |\mathcal{A}|$, still a polynomial. We still need to check whether this guessed plan is also executable, but this can be done in linear time. This suffices to show that the problem is thus in NP.

Hardness: We reduce the NP-complete *$diMAPF^{dag,uc}$* (Theorem 4) to *$diMAPF^{dag,uc}_{\langle ms,B \rangle}$* by using the exact same reduction as for Corollary 5. This time we do not even need the logarithmic encoding since we know that plans for *$diMAPF^{dag,uc}$* problems are bounded by the polynomial $|V| \cdot |\mathcal{A}|$, so we use $B = |V| \cdot |\mathcal{A}|$ thus completing the construction as clearly *$diMAPF^{dag,uc}$* admits a solution of bound B if and only if *$diMAPF^{dag,uc}$* has any solution at all. □

Results obtained up to here followed relatively directly from the literature. We move on in the subsequent two sections posing additional structural restrictions on the problem for which we can still maintain NP-completeness.

4.2 OdP with Makespan Restrictions

We are interested in exploring the extent to which the makespan, denoted as B, can be constrained while the problem remains NP-hard. To initiate this investigation, we begin by imposing a polynomial restriction on the number of nodes in the graph associated with the problem.

Now, let us consider an additional scenario in which the graph involved in the given problem is both acyclic and unilaterally-connected. Under these conditions, the problem, formally denoted as *$diMAPF^{dag,uc}_{\langle ms,|V| \rangle}$*, can be shown to be NP-complete. That is, we now have a special case of the previous *$diMAPF^{dag,uc}_{\langle ms,B \rangle}$*, so membership follows directly. Meanwhile, to prove the hardness part, we can reduce from *$diMAPF^{dag,uc}_{\langle ms,B \rangle}$*. Specifically, we only need to make sure that it remains solvable within a makespan that equals the number of nodes of the original problem. Now, if the bound B happens to be equal to or smaller than the number of nodes in the input problem, nothing has to be done. If it is however larger (with a possible maximum of $|V| \cdot |A|$), we increase the number of nodes to this number, but in a way that the graph remains unilaterally connected, and that the additional vertices cannot be used – in the same way as in the previous proof for Theorem 4. As we only need polynomially many new vertices and agents, the transformation runs in poly-time.

Also note that, from Theorem 1 of [9], our Corollary 5, Corollary 6, and Theorem 2, it is also straightforward to obtain the NP-completeness result for the problem of *$diMAPF^{dag,pl}_{\langle ms,|V| \rangle}$*, where the graph is acyclic and planar.

We now move on to further restricting the bound B. More specifically, we are interested in finding a lower bound (*lb*) for the actual value of B, such that when "$B < lb$" holds, *$diMAPF^{\mathcal{R}}_{\langle O,B \rangle}$* can not admit any solution anyway, as B is too small. The lower bound for *ms* is first investigated.

Definition 5 *Given a $diMAPF^{\mathcal{R}}_{\langle ms,lb_{ms} \rangle}$ problem, lb_{ms} is a tight lower bound makespan, in the sense that, there does not exist a $\vec{\mathcal{S}}$, which is a solution to $diMAPF^{\mathcal{R}}$, and $|\Sigma| < lb_{ms}$. In other words, for any integer $n > 0$, the problem $diMAPF^{\mathcal{R}}_{\langle ms,(lb_{ms}-n) \rangle}$ does not admit a solution.*

Definition 6 (LB_{ms} of $diMAPF^{\mathcal{R}}$) *Given a $diMAPF^{\mathcal{R}}$ problem, a set $\mathcal{P}_{min}$ containing the shortest paths from their initial vertices to their goal vertices, for all the agents in the problem, and suppose $\hat{p} \in \mathcal{P}_{min}$ is a path whose length is the maximal one among all paths in the set, we define that $LB_{ms} = |\hat{p}|$.*

Proposition 2 *Given a $diMAPF^{\mathcal{R}}$,*
- *$\mathcal{P}_{min}$, thus LB_{ms}, can be calculated in poly-time; And*
- *given $diMAPF^{\mathcal{R}}_{\langle ms,LB_{ms} \rangle}$, LB_{ms} is actually an instance of lb_{ms}.*

Proof The set $\mathcal{P}_{min}$ can be constructed through running Dijkstra's algorithm on the graph G in *$diMAPF^{\mathcal{R}}$*, for each agent in the problem, to find out the shortest path for the agent to move from its initial vertex to the goal one.

From $\mathcal{P}_{min}$, we obtain LB_{ms}. Even if it is possible for all agents non-stop moving between steps, it still takes at least as many as LB_{ms} steps to arrive at all their respective goal vertices. Hence the makespan value can not be smaller than LB_{ms}. In other words, LB_{ms} is a tight lowerbound for makespan. □

The following theorem however indicates that even if we are dealing with this boundary case on the B value, the problem becomes NP-hard.

Theorem 7 *diMAPF*$^{dag,pl}_{\langle ms,LB_{ms}\rangle}$ *is NP-complete.*

Proof The problem as a further restricted version to an NP-complete problem, remains in NP.

In order to prove the NP-hardness of the problem, we can use the same construction in the proof for Theorem 2. Note that in the constructed graph the longest path among all agents is the one for the B-agent. Suppose originally there are n variables and k clauses, then the length of this longest path is $3n+k+1$. In Figure 4, for example, the length is thus $3 \cdot 4 + 3 + 1 = 16$.

It is clear that a *RPMS-3SAT* instance is satisfiable iff the resulting *diMAPF*$^{dag,pl}_{\langle ms,LB_{ms}\rangle}$ instance $\langle G_{(dag,pl)}, \mathcal{A}, \mathcal{I}, \mathcal{G}\rangle$ admits a solution within $3n + k + 1$ steps. □

The intractability result can be extended into the case where we know of the existence of "**sol**", which is a solution not subject to the makespan constraint. In other words, "sol" is provided as additional input.

Corollary 8 *diMAPF*$^{dag,pl}_{\langle ms,LB_{ms}\rangle,sol}$ *is NP-complete.*

Proof The problem is again in NP, as it is a special case to the NP-complete *diMAPF*$^{dag,pl}_{\langle ms,LB_{ms}\rangle}$ (proved in the theorem above).

To prove it is NP-hard, again we start up from the construction in the proof for Theorem 2. We will introduce into the graph bridge-paths, which are too long in length thus can not be used when the makespan constraint is applied. Precisely, each bridge-path will connect two existing vertices, with more than $3n + k + 1$ vertices in between. One bridge-path connects from B to B^g. Within each diamond, there is also a bridge-path connecting the left vertex with the right one. Hence we know that without the makespan constraint, the B-agent can simply take its bridge-path to arrive at its goal. At the same time, all v-agents will use their corresponding bridge-paths to their goals too. Hence all c-agents will never be blocked by any v-agents. As long as they follow the right order from c_1 to c_k, they can always arrive at their goal vertices. This arrangement corresponds to a solution for the problem, which is not subject to the "*LB*$_{ms}$" constraint. However, it is still NP-hard to decide the existence of a solution that is bounded by $3n + k + 1$, equaling to the length of the shorted path for the B-agent. □

4.3 OdP with Restrictions on Path Length

The availability of Theorem 2 to Corollary 8 allows us to perform complexity analysis on optimal *diMAPF* problems on the optimization criteria related to travel distance of agents, relatively easily. The definitions are given below first.

Definition 7 *We first identify respectively in the following two lower bounds for maximal individual travel distance, and for total distance.*

- ***(lower bound of maximal distance, written as "$lb_{\max}$")*** *We can again in polynomial-time find out $\mathcal{P}_{min}$ the shortest paths for all the agents in a given problem instance. Among them, we pick up $\hat{p}$ the longest one, whose length $|\hat{p}| = LB_{\max}$, which serves as the lower bound for B (justification is similar to the one for lb_{ms}/LB_{ms}).*
- ***(lower bound of total distance, written as "lb_{tot}")*** *We see that the actual lb_{tot} should be instantiated into $LB_{tot} = \sum_{p\in\mathcal{P}_{min}} |p|$. Any path $p \in \mathcal{P}_{min}$ is the shortest one for its agent, the sum of all path lengths in any solution to the given problem thus can not be shorter than LB_{tot}.*

The following theorem is a direct consequence of Theorem 7, where the B-agent moves without a stop between steps in the proof for NP-hardness, implying that LB_{ms} and LB_{max} having the same value $3n + k + 1$.

Theorem 9 *diMAPF*$^{dag,pl}_{\langle \max,LB_{\max}\rangle}$ *is NP-complete.*

Regarding the lower bound of total distance, i.e., lb_{tot}, a little bit more is involved to prove the NP-completeness. An idea for such a proof is given below, using the same example transformation as before.

Theorem 10 *diMAPF*$^{dag,pl}_{\langle tot,LB_{tot}\rangle}$ *is NP-complete.*

Proof The problem as a further restricted version to an NP-complete problem, remains in NP.

In order to prove the NP-hardness of the problem, we can simply revise the construction in the proof for Theorem 2. Note again that in the original proof, the path length for the B-agent is the longest one equaling to $3n+k+1$, where n is the total number of variables, and k is the total number of clauses in the *RPMS-3SAT* instance. The path length for each v-agent is exactly 2. However, the path lengths for c-agents differ from each other. Even for a given c-agent, the actual path length varies, depending on which one of the three legs the agent will take to reach its goal vertex.

In the revision, however, we ask the path length for any c-agent traversing through any one of its three legs to be $3n + k + 1$, always. This can be easily achieved by inserting an appropriate number of intermediate vertices on the leg-edges. Taking the c-agent c_2 in Figure 4 for example (note that the path length for the B-agent is $3 \cdot 4 + 3 + 1 = 16$). The original path lengths traversing through three different legs are 10, 7 and 4, respectively. We can insert accordingly 6, 9 and 12 intermediate vertices into the left, middle, and right legs respectively. As such, should c_2 reach its goal vertex c_2^g, the path would always be $10 + 6 = 7 + 9 = 4 + 12 = 16$.

Hence, a *RPMS-3SAT* instance is satisfiable iff the resulting *diMAPF*^{dag,pl} instance $\langle G_{(dag,pl)}, \mathcal{A}, \mathcal{I}, \mathcal{G}\rangle$ admits a solution with overall travel distance for all agents bounded by $(k + 1)(3n + k + 1) + 2n = 3nk + k^2 + 5n + 2k + 1$ steps[5]. □

We can again add the constraint on the degree of vertices (i.e., d $\leq$ 3) into the problem. That is, we can add up to k intermediated vertices before entering these nodes such that the d $\leq$ 3 constraint is further satisfied (see again Figure 3.(b) for illustration). The complexity results also can be extended to sol. We have the following result (actual proof skipped).

Theorem 11 *All three of the following problems are NP-complete:* *diMAPF*$^{dag,pl,d\leq 3}_{\langle ms,LB_{ms}\rangle,sol}$, *diMAPF*$^{dag,pl,d\leq 3}_{\langle \max,LB_{\max}\rangle,sol}$ *and diMAPF*$^{dag,pl,d\leq 3}_{\langle tot,LB_{tot}\rangle,sol}$.

5 Discussion and Conclusion

Computational complexity of MAPF on directed graphs (i.e., *diMAPF*) was only recently studied [9, 10]. This paper investigates the complexity of optimal *diMAPF* problems. That is,

1. the *diMAPF* problems are further subject to the optimization criterion of minimizing either the makespan for all agents, the maximal travel distance for any agent, or the overall travel distance for all agents; Meanwhile,

[5] It is interesting to observe that we can simply use the transformation in this current theorem to prove the NP-hardness part in both Theorem 7 and Theorem 9, with minimal efforts involved. After all, these three theorems deal with three NP-complete problems, which are closely related to each other.

2. a variety of different special cases of the problem are considered. For example, the digraphs are restricted to be planar, or/and are with bounded vertex-degrees.

In particular, relationship between the results of [9, 10] and the ones in this paper, is established by Theorem 1, Corollaries 5 and 6.

We identified a general intractability (i.e., NP-hardness) of all these sub-problems. Nevertheless it is surprising to note that, for each one of these three criteria, even if it is applied with the smallest-possible value, optimal *diMAPF* remains to be NP-hard (e.g., Theorems 7, 9, and 10, for the criteria of makespan, maximal travel distance, and total travel distance, respectively).

Theorem 2 is our primary result, where digraphs are restricted to planar ones. Several more results easily follow. The availability of these complexity results, which are related to planar graphs, in our opinion has important real-world implications. For instance, we now know that when on a two-dimensional space only (**pl**), with acyclic one-way roads only (**dag**), and at most three-branch junctions only ($\mathbf{d} \leq \mathbf{3}$), multi-agent path-finding is already computationally challenging (Theorem 11, in fact, regardless whether or not the problem is subject to the makespan constraint).

Many real-world MAPF problems are indeed solvable if left unconstrained – just allow one robot/agent to move at any given step. However those theorems, where a solution "**sol**" is provided as additional input, indicate that actual computational challenge lies in whether the agents can arrive within the makespan (Theorems 8 and 11). In general, results in this paper should be helpful in pinpointing sources contributing to the intractability, thus in guiding us for development of better heuristics/algorithms for these problems. For example, it would be interesting to study how these results are related to the phenomenon of "pairwise symmetry" leading to collisions [3, 8].

For future work, while we know that the decision version of general MAPF is poly-time solvable but the general decisional *diMAPF* is NP-hard [7, 9], we do not know whether a similar difference could be observed in any special case between the optimization version of *MAPF* and the one of *diMAPF*. For example, we have Corollary 11, but we do not know whether or not $MAPF^{dag,pl,\mathrm{d}\leq 3}_{\langle ms, LB_{ms}\rangle, sol}$ remains NP-complete, or is poly-time solvable. The closest result so far, to our best knowledge, is that optimal *MAPF* on a planar graph is NP-hard [21, 20].

Acknowledgements

We would like to thank the referees for dedicating their time to read our paper and for providing us with valuable feedback. Their constructive input has contributed to the improvement of this paper.

References

[1] Jacopo Banfi, Nicola Basilico, and Francesco Amigoni, 'Intractability of time-optimal multirobot path planning on 2d grid graphs with holes', *IEEE Robotics and Automation Letters*, **2**(4), 1941–1947, (2017).

[2] Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, David Tolpin, Oded Betzalel, and Eyal Shimony, 'ICBS: Improved conflict-based search algorithm for multi-agent pathfinding', in *Proceedings of the 25th IJCAI*, pp. 740–746, (2015).

[3] Zhe Chen, Daniel Damir Harabor, Jiaoyang Li, and Peter J. Stuckey, 'Symmetry breaking for k-robust multi-agent path finding', in *Proceedings of the 35th AAAI*, pp. 12267–12274, (2021).

[4] Mark de Berg and Amirali Khosravi, 'Optimal binary space partitions in the plane', in *Computing and Combinatorics*, eds., My T. Thai and Sartaj Sahni, pp. 216–225, Berlin, Heidelberg, (2010). Springer Berlin Heidelberg.

[5] Tzvika Geft and Dan Halperin, 'Refined hardness of distance-optimal multi-agent path finding', in *Proceedings of the 21st AAMAS*, pp. 481–488, (2022).

[6] Dan Halperin, J-C Latombe, and Randall H Wilson, 'A general framework for assembly planning: The motion space approach', *Algorithmica*, **26**(3), 577–601, (2000).

[7] D. Kornhauser, G. Miller, and P. Spirakis, 'Coordinating pebble motion on graphs, the diameter of permutation groups, and applications', in *Proceedings of the 25th FOCS*, pp. 241–250, (1984).

[8] Jiaoyang Li, Daniel Harabor, Peter J. Stuckey, Hang Ma, Graeme Gange, and Sven Koenig, 'Pairwise symmetry reasoning for multi-agent path finding search', *Artificial Intelligence*, **301**, 103574, (2021).

[9] Bernhard Nebel, 'On the computational complexity of multi-agent pathfinding on directed graphs', in *Proceedings of the 30th ICAPS*, pp. 212–216, (2020).

[10] Bernhard Nebel, 'The small solution hypothesis for MAPF on strongly connected directed graphs is true', in *Proceedings of the 33rd ICAPS*, pp. 304–313, (2023).

[11] Keisuke Okumura, François Bonnet, Yasumasa Tamura, and Xavier Défago, 'Offline time-independent multi-agent path planning', in *Proceedings of the 31st IJCAI*, pp. 4649–4656, (2022).

[12] Oren Salzman and Dan Halperin, 'Asymptotically near-optimal rrt for fast, high-quality motion planning', *IEEE Transactions on Robotics*, **32**(3), 473–483, (2016).

[13] Oren Salzman and Roni Stern, 'Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems', in *Proceedings of the 19th AAMAS*, pp. 1711–1715, (2020).

[14] David Šišlák, Přemysl Volf, and Michal Pěchouček, 'Agent-based cooperative decentralized airplane-collision avoidance', *IEEE Transactions on Intelligent Transportation Systems*, **12**(1), 36–46, (2010).

[15] Trevor Standley, 'Finding optimal solutions to cooperative pathfinding problems', in *Proceedings of the 24th AAAI*, pp. 173–178, (2010).

[16] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski, 'Multi-agent pathfinding: Definitions, variants, and benchmarks', in *Proceedings of the 12th SoCS, 2019*, pp. 151–159, (2019).

[17] Pavel Surynek, 'An optimization variant of multi-robot path planning is intractable', in *Proceedings of the 24th AAAI*, pp. 1261–1263, (2010).

[18] Jindrich Vodrázka, Roman Barták, and Jirí Svancara, 'On modelling multi-agent path finding as a classical planning problem', in *32nd IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2020, Baltimore, MD, USA, November 9-11, 2020*, pp. 23–28. IEEE, (2020).

[19] Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz, 'Coordinating hundreds of cooperative, autonomous vehicles in warehouses', *AI Magazine*, **29**(1), 9–20, (2008).

[20] Jingjin Yu, 'Intractability of optimal multirobot path planning on planar graphs', *IEEE Robotics and Automation Letters*, **1**(1), 33–40, (2016).

[21] Jingjin Yu and Steven M. LaValle, 'Structure and intractability of optimal multi-robot path planning on graphs', in *Proceedings of the 27th AAAI*, pp. 1443–1449, (2013).