Inapproximability of Optimal Multi-Agent Pathfinding Problems

Xing Tan[1], **Alban Grastien**[2]

[1]Department of Computer Science, Lakehead University, Canada
[2]Université Paris-Saclay, CEA, List, F-91120, Palaiseau, France
xing.tan@lakeheadu.ca, alban.grastien@cea.fr

Abstract

Multi-agent pathfinding (*MAPF*) is a problem where multiple autonomous agents must find paths to their respective destinations without colliding. Decisional *MAPF* on undirected graphs can be solved in polytime; several optimization *MAPF* variants however are *NP*-complete. The directed graph variant (*diMAPF*) is more complex, with its decisional version already being *NP*-complete. This paper examines the computational approximability of optimal *MAPF* problems (i.e., minimizing makespan for agent travel distance and maximizing the total number of agents reaching their goals), providing a first set of several inapproximability results for these problems. The results reveal an inherent limitation in approximating optimal solutions for *MAPF*s, provide a deeper understanding regarding their computational intractability, thus offer foundational references for future research.

1 Introduction

Multi-agent pathfinding (*MAPF*) is a problem in which multiple autonomous agents, such as robots in a warehouse, navigate paths to their individual destinations while avoiding collisions with one another (Wurman, D'Andrea, and Mountz 2008; Stern et al. 2019). This problem has garnered significant attention in both research and practical applications, given its relevance to various fields such as robotics, operations research, and AI planning (Salzman and Stern 2020; Okumura et al. 2022). For example, in robotics, *MAPF* is essential for coordinating autonomous drones or mobile robots in factories; in operations research, it plays a vital role in optimizing the transportation of goods and personnel in logistics and supply chain management (Salzman and Halperin 2016; Boyarski et al. 2015; Vodrázka, Barták, and Svancara 2020; Standley 2010; Šišlák, Volf, and Pěchouček 2010).

Determining whether a solution exists for a *MAPF* problem can be done efficiently in polytime for undirected graphs (Kornhauser, Miller, and Spirakis 1984). However, optimizing *MAPF* problems, such as finding the optimal makespan or travel distance for the agents, is *NP*-complete on general graphs and various constrained settings, like planar graphs or grids (Surynek 2010; Yu and LaValle 2013; Yu 2016; Banfi, Basilico, and Amigoni 2017; Geft and Halperin 2022). The study of *MAPF* on directed graphs (*diMAPF*) is relatively recent, with findings indicating that both of its decisional and optimal variants are *NP*-complete (Nebel 2020, 2023; Tan and Bercher 2023; Nebel 2024).

Given the intractability of optimal *MAPF* problems, current research often emphasizes finding suboptimal solutions that balance practical efficiency and solution quality. Aiming to provide feasible solutions within reasonable computational limits, these algorithms focus on minimizing the makespan (de Wilde, ter Mors, and Witteveen 2013; Barer et al. 2014; Cohen et al. 2016), and/or maximizing the number of agents reaching their goals (Ma et al. 2018). We believe however that, in the development and application of these approximate algorithms for *MAPF*, delineating the boundaries of what is computationally feasible is important. Knowing the exact limitations and inherent difficulties of the problem helps set realistic expectations and design strategies that work within given constraints. Furthermore, intractability results can guide the development of heuristics and approximation algorithms, ensuring that they are both effective and efficient.

This paper presents, to the best of our knowledge, a first set of inapproximability results concerning the optimal makespan and the total number of agents reaching their goals. In Section 3, we examine the complexity of *MAPF*/*diMAPF* which are constrained on maximally-allowed makespan. We investigate problems of maximizing the total number of agents reaching their goals with a makespan of three agent-steps at most, proving their non-approximability, that is, non-existence of any polytime algorithm solving these problems sub-optimally within a certain factor ($\frac{662}{665} \approx 0.995$). We then in Section 4 examine the complexity of *MAX-diMAPF*, i.e., the optimal variant of *diMAPF*, on maximizing the total number of agents reaching their goals. We prove for this problem, the non-existence of any polytime algorithm within a certain factor (initially $\frac{113}{120} \approx 0.942$, and later the bound is refined to $\frac{57}{64} \approx 0.891$). Our findings reveal the inherent limitations in approximating optimal solutions for *MAPF*, providing a deeper understanding of the problem's complexity, thus offering foundational references for future research.

The remainder of this paper is organized as follows. Section 2 covers the preliminaries. Our complexity analysis is detailed in Sections 3 and 4. Section 5 concludes the paper.

2 Preliminaries

This section provides preliminary knowledge essential for the paper. It begins with definitions related to *MAPF*. Following this, it covers the basics of complexity analysis, with a particular focus on inapproximability analysis techniques. Finally, it introduces two relevant *NP*-complete problems, optimal *SAT* and optimal *3DM*, which are used in the proofs for the inapproximability results.

2.1 MAPF Problems

We formally define decisional *MAPF* and *diMAPF* problems as well as their optimal variants (e.g., bounded on makespan). Definitions follow the ones by Nebel in (Nebel 2020, 2023). 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. 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.

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 E 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 $\mathcal{S}_0$ 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 $\mathcal{S}_i$. Let $\mathcal{S}_0 \equiv \mathcal{I}$ be the initial state, and suppose that $\mathcal{S}_n \equiv \mathcal{G}$ is actually the final state where all the agents are injectively mapped into their respective goal-vertices. 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 $\mathcal{P}_\Sigma$. 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.

[1]Some works (cf., e.g., (Stern et al. 2019)) assume that an agent will stay and will never move again after it reached its goal – thus preventing any other agent from passing through it in future steps. We allow agents to move away from their goal vertex.

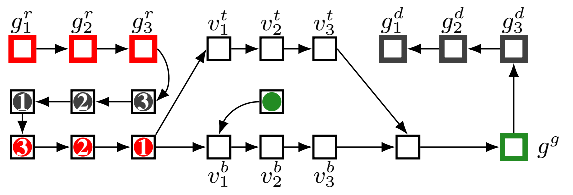

Figure 1: A *MAX-diMAPF* example. Squares represent vertices and filled circles depict agents. Agents are grouped by different colors to illustrate their interactions. Each agent has a unique goal vertex. An agent's goal is named g and identified by the sub- and superscript (e.g., g_2^d is the goal of the second dark agent). A goal vertex is highlighted with thick lines matching the color of its agent. The maximum number of arrivals is 4: Three for the dark agents and one for the green agent. The three red agents will never reach their goals.

Definition 1 (*MAPF/diMAPF*)**.** *A multi-agent pathfinding problem instance is a four-tuple $\langle G, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$, where G is a directed (or undirected) graph, $\mathcal{A}$ is a set of agents, $\mathcal{I}$ is the initial state, and $\mathcal{G}$ is the goal state. Is there a sequence Σ that moves all agents in $\mathcal{A}$ from $\mathcal{I}$ to $\mathcal{G}$?*

A natural optimal variant to decisional *MAPF* is to find out the optimal number of agents that can be moved to their goal-vertices. Note that this variant relaxes the requirement that, in $\mathcal{S}_n$, all agents are in their respective goals.

Definition 2 (*MAXMAPF/MAX-diMAPF*)**.** *Given a MAPF/diMAPF problem instance $\langle G, \mathcal{A}, \mathcal{I}, \mathcal{G} \rangle$, where G is a directed (or undirected) graph, what is the largest number of agents a such that $\mathcal{S}_n(a) = \mathcal{G}(a)$ when applying a sequence Σ?*

An example *MAX-diMAPF* is given in Figure 1[2]. It can be verified that the maximum number of agents arriving to their respective goals is 4. The three red agents (r-agents) will never reach their goals (g_1^r–g_3^r) in this acyclic graph. The three dark agents (d-agents), on the other hand, can reach their goals (g_1^d–g_3^d) if they push the r-agents out of the way either to the top vertices (v_1^t–v_3^t), or to the bottom ones (v_1^b–v_3^b). For the green agent (g-agent) to also reach its own target g^g without blocking the dark agents, the r-agents would have to take the top path.

We now consider optimal makespan.

Definition 3 (*MAPF*$_{\langle O,B \rangle}$ or *diMAPF*$_{\langle O,B \rangle}$)**.** *MAPF$_{\langle O,B \rangle}$, or diMAPF$_{\langle O,B \rangle}$, is a MAPF (or diMAPF) problem subject to an optimization criterion of type O, which is bounded by $B \in \mathbb{Z}^+$. When $O = ms$, it means to answer: Is there a MAPF (or diMAPF) solution Σ such that $|\Sigma| \leq B$?*

Finally, we define the optimal *MAPF/diMAPF*, on maximizing agents reaching their goals with makespan bounded by the value *B*.

[2]The example of Figure 1 illustrates how decisions – whether to send the r-agents to the top or to the bottom – can be encoded in a *diMAPF*, a critical property that will be used when proving Proposition 3.

Definition 4 (*MAXMAPF*$_{\langle ms,B\rangle}$/*MAX-diMAPF*$_{\langle ms,B\rangle}$)**.** *Given a MAPF/diMAPF problem instance $\langle G, \mathcal{A}, \mathcal{I}, \mathcal{G}\rangle$, where G is an undirected (or directed graph), how many agents in $\mathcal{A}$ from $\mathcal{I}$ can arrive at $\mathcal{G}$, within makespan B?*

2.2 Inapproximability Theory

This section provides standard preliminaries on inapproximability theory (Papadimitriou and Yannakakis 1991).

Definition 5. *Let α be a ratio, called an optimization ratio. An α-approximation algorithm for an optimization problem is an algorithm that returns a solution within an α factor of the optimal solution in polytime.*

We use the convention that $\alpha \geq 1$ for minimization problems, and $\alpha \leq 1$ for maximization problems. For example, a $\frac{2}{3}$-approximation algorithm for a maximization problem is a polytime algorithm that returns a solution for any instance of the problem, and the solution is at least $\frac{2}{3}$ to the optimal solution for the instance. A $\frac{3}{2}$-approximation algorithm for a minimization problem is a polytime algorithm that returns a solution at most $\frac{3}{2}$ to the optimal solution for the instance.

Definition 6 (L-reduction with parameters a and b)**.** *Two positive numbers $a > 0$ and $b > 0$, and two maximization problems Π and Π' are given. A reduction from Π to Π' is an L-reduction (also called as Linear reduction, or approximation-preserving reduction), if the following conditions are satisfied*

1. *For any instance I of Π, an instance I' of Π' can be computed in polytime through this reduction.*
2. *Let OPT(I) be the value of a maximum solution to I of Π, and OPT(I') the one for the constructed I' of Π', the inequality OPT$(I') \leq a \cdot$ OPT(I) holds.*
3. *For any solution of value V' to I', a solution of value V to I can be computed accordingly in polytime such that*

$$|OPT(I) - V| \leq b \cdot |OPT(I') - V'|.$$

Theorem 1. *(Papadimitriou and Yannakakis 1991) Given two maximization problems Π and Π', if 1) there exists an L-reduction (with parameters a and b) from Π to Π', and 2) there is an α-approximation algorithm for Π', then there exists an $(a \cdot b \cdot (\alpha - 1) + 1)$-approximation algorithm for Π. In other words, if it is NP-hard to approximate Π within a factor of $(a \cdot b \cdot (\alpha - 1) + 1)$, there does not exist an α-approximation algorithm for Π', unless P = NP.*

2.3 Maximal SAT and Maximal 3DM

We will perform some reductions from the *MAXE3SAT*, the maximum satisfiability problem where each clause has exactly three literals. An example (with $n = 4$ variables and $m = 5$ clauses) is $(x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$. Of course, the example is satisfiable.

We have the following well-known result for *MAXE3SAT*.

Theorem 2. *(Johnson 1974; Yannakakis 1994; Håstad 2001) For any instance of MAXE3SAT, there exists a truth assignment that satisfies at least $\frac{7}{8}$ of the clauses. If there exists a $(\frac{7}{8} + \epsilon)$-approximation algorithm for the MAXE3SAT problem for any constant $\epsilon > 0$, then P = NP.*

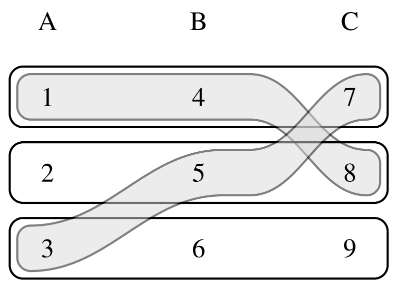

Figure 2: A *3DM* instance. Three disjoint sets in three columns, each containing $m = 3$ elements, $A = \{1, 2, 3\}$, $B = \{4, 5, 6\}$, $C = \{7, 8, 9\}$. A set of $n = 5$ triples $\mathcal{T} = \{T_1, T_2, \ldots, T_5\}$, where $T_1 = \{1, 4, 7\}$, $T_2 = \{2, 5, 8\}$, $T_3 = \{3, 6, 9\}$, $T_4 = \{1, 4, 8\}$, $T_5 = \{3, 5, 7\}$. The set $\mathcal{S} = \{T_1, T_2, T_3\}$ forms a matching of $\mathcal{T}$.

Some reductions in the paper need to use the *NP*-complete 3-dimensional matching (*3DM*) problem (Karp 1972).

Definition 7 (*3DM* problem)**.** *A 3DM problem instance is a tuple $\langle A, B, C, \mathcal{T}\rangle$ where $A = \{a_1, a_2, \ldots, a_m\}$, $B = \{b_1, b_2, \ldots, b_m\}$, and $C = \{c_1, c_2, \ldots, c_m\}$ are three disjoint sets of cardinality m and $\mathcal{T}$ is a set of n triples $\mathcal{T} = \{T_1, T_2, \ldots, T_n\} \subseteq (A \cup B \cup C)$ such that each triple $T_i \in \mathcal{T}$ contains exactly one element from A, B, and C, respectively. A partial matching of $\mathcal{T}$ is a subset $S \subseteq \mathcal{T}$ of disjoint elements (i.e., $\forall\, T_i, T_j \in S,\ T_i \neq T_j \Rightarrow T_i \cap T_j = \emptyset$). The matching is complete if $|S| = m$ or, equivalently, $\bigcup_{T \in S} T = A \cup B \cup C$. The 3DM problem asks the question: Does $\mathcal{T}$ contain a (complete) matching?*

The *3DM* problem is *NP*-complete, and remains so even if no element occurs in more than three triples (Garey and Johnson 1979). An example *3DM* is given in Figure 2. The example is also used as the running example to explain the polytime reduction.

MAX3DM is an optimal variant to *3DM*, defined below.

Definition 8 (*MAX3DM*)**.** *Given an instance of 3DM, find $\mathcal{S}$, which is a maximal partial matching of $\mathcal{T}$, and $|S| = k$, the size of S. That is, among all partial matchings of $\mathcal{T}$, S is maximal in terms of set size.*

Theorem 3. *(Cygan 2013) For any instance of MAX3DM, there exists a polytime $\frac{3}{4}$-approximation algorithm for the instance.*

Theorem 4. *(Chlebík and Chlebíková 2006) It is NP-hard to approximate MAX3DM within a factor of $\frac{94}{95}$, even for instances where each element has exactly two occurrences in the triples.*

3 Inapproximability of *MAPF*s on Makespan

In this section, we examine the complexity of *MAPF* (on both directed and undirected graphs) constrained on makespan. We first propose a reduction from *3DM* to *diMAPF*, laying the foundation for the subsequent results. Specifically, we demonstrate the *NP*-completeness of *MAPF* with a makespan of 3, thus establishing the non-polytime approximability for any makespan less than 4. We then consider a particular case where the makespan can be reduced to 2 while maintaining all intractable results. Finally, we address the problem of maximizing the total number of agents reaching their goals within a makespan of 3, proving the non-approximability of any polytime algorithm within a certain factor.

Reduction We use Figure 3 to illustrate the reduction. Given an instance of *3DM* problem with $3m$ elements and n triples, we construct a *diMAPF* problem with $4m$ vertices, m agents, $3n$ edges, plus m "dashed-boxes" (explained later). For each B element that appears $1 + k$ times in $\mathcal{T}$, $4k$ vertices, k agents, and $5k$ edges need to be added (an example with $k = 1$ is shown in Figure 4). The transformation is polytime. In fact, assuming an element appears no more than 3 times (Garey and Johnson 1979, *3DM* still *NP*-complete), $k \leq 2$ holds.

More precisely, for each of the three sets of A, B, and C, we introduce accordingly three layers. Layers A and C each contains m vertices. Layer B contains m dashed-boxed. One additional Layer, D, containing another m vertices is also introduced. We use the set $\mathcal{T}$ to guide us to introduce directed-edges from Layers A to B, and from B to C. That is, if $T_i = \{a, b, c\} \in \mathcal{T}$, then edges (a, b) and (b, c) are introduced in $\mathcal{G}$. Multi-edges are possible. For example, in Figure 3, there are two edges between vertices 1 and 4, as elements 1 and 4 occur in both T_1 and T_4. Vertices in Layer D are primed versions of those in Layer A, and for each triple $T_i = \{a, b, c\} \in \mathcal{T}$, an edge is created from c to a'.

Let us now describe the dashed-boxes.

The goal of the reduction is to ensure that if an agent successfully travels from vertex a of Layer A to vertex a' of Layer D in 3 steps, the path $a \to b \to c \to a'$ it takes will correspond to a triple $T_i = \{a, b, c\}$ from $\mathcal{T}$. To this end, for each such triple T_i, a vertex b_i is created with two edges (a, b_i) and (b_i, c): each path $a \to b_i \to c \to a'$ can be linked to T_i. However, we need to ensure that no two agents travel through b_i and b_j (from two paths linked to T_i and to T_j, respectively, such that $b \in T_i \cap T_j$). For this purpose, we introduce additional agents.

If $k+1$ is the number of triples that contain b, we create k additional agents that each need to travel through one of the b_i in order to reach its goal vertex (cf. Figure 4).

Properties of the reduction Assuming a makespan of 3, all agents have to move towards their goal at each step. This means that the path $a \to b_i \to c \to a'$ of a non-dashed-box agent corresponds to the triple $T_i = \{a, b, c\} \in \mathcal{T}$. Furthermore, because each dashed-box agent also reaches its respective goal in 3 steps, element b can be selected by at most one non-dashed-box agent (and for a solution exactly one, since there are m agents for m elements in B). Thus, each solution to the *diMAPF*$_{\langle ms, B \leq 3 \rangle}$ can be mapped to a solution to the original *3DM* problem, and conversely.

Theorem 5. *The problem diMAPF*$_{\langle ms, B \leq 3 \rangle}$*, i.e., multi-agent pathfinding on a directed graph with makespan bounded by 3 steps, is NP-complete.*

Proof. The problem is in *NP* because it is a restricted variant of *NP*-complete *diMAPF* (Nebel 2023). The problem is *NP*-hard because of the described reduction. □

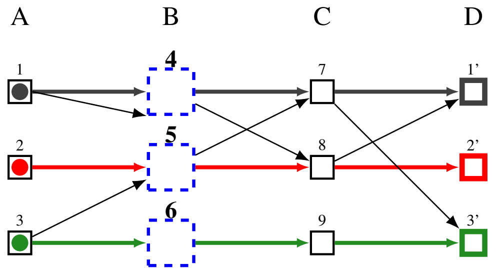

Figure 3: The *diMAPF*$_{\langle ms, B \leq 3 \rangle}$ instance, constructed from the given example *3DM* instance (Figure 2), introduces one vertex for each number in Figure 2, labeled accordingly. The vertices are organized into three layers: A, B, and C, with an additional Layer D. There are three agents: dark, red, and green. All agents initially start at Layer A, with their respective goal vertices located in Layer D. Layer B contains dashed-boxes. An inside look of a dashed-boxes (e.g., Dashed-box 5) is presented in Figure 4.

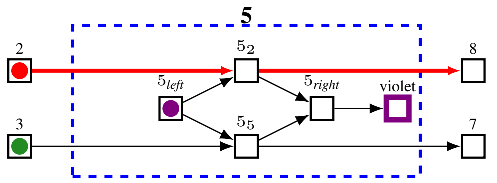

Figure 4: An inside look of Dashed-box 5. If the two agents follow the paths $2 \to 5 \to 8$ of $T_2 = \{2, 5, 8\}$ and $3 \to 5 \to 7$ of $T_5 = \{3, 5, 7\}$, then the violet agent will not be able to reach its goal in 3 steps.

We get the following result, which is an immediate corollary of Theorem 5.

Corollary 6. *For any $\alpha < 4/3$, there does not exist a polytime α-approximation algorithm for minimizing makespan of a diMAPF problem, unless P = NP.*

Observe that in the above reduction, all the agents must move non-stop to meet the make-span constraint. They will have no opportunity to move backward, even if the edges are undirected. This observation allows us to extend the above results for *MAPF* problems. We get the following result (Ma et al. 2016).

Corollary 7. *(Ma et al. 2016, Corollary 8) The problem MAPF*$_{\langle ms, B \leq 3 \rangle}$*, i.e., multi-agent pathfinding on an undirected graph with makespan bounded by 3 steps, is NP-complete. For any $\alpha < 4/3$, there does not exist a polytime α-approximation algorithm for the problem, unless P = NP.*

Nevertheless, our approach of using *3DM* for the reduction enables us to obtain some additional results that are theoretically closer to the tractability dichotomy.

Corollary 8. *When agents/goals are typed (i.e., any goal is for any agent with the same type), and when there are only* *two types of agents/goals, both the problems MAPF$_{\langle ms,B\leq 2\rangle}$ and diMAPF$_{\langle ms,B\leq 2\rangle}$, i.e., multi-agent pathfinding on a directed or undirected graph with makespan bounded by 2 steps, are NP-complete. Consequently, for any $\alpha < 3/2$, there does not exist a polytime α-approximation algorithm for the problem, unless P = NP.*

Proof. We can simply refine the proof for Theorem 5. Now we have only two types, violet, and the other "white" ones. Specifically we 1) remove completely Layer D and all the edges entering the vertices on Layer D, and 2) make all the vertices on Layer C as goal vertices for all the agents at Layer A, which means these agents initially at Layer A are now white. Further for the gadget part, as the makespan requirement is now 2, there would be just one right-vertex, which is also a goal vertex for the v-agent.

Each white agent (i.e., w-agent) can take any white goal (i.e., w-goal). Since there are exactly m w-agents, and m w-goals, after two steps, each agent would take exactly one w-goal, should a solution exist. This NP-completeness result with makespan 2 enables us to conclude that there does not exist a polytime $\frac{3}{2}$-approximation algorithm for the problem, unless *P* = *NP*. □

Proposition 1. *Given any MAX3DM*[3] *instance I with three disjoint sets each containing m elements, and a set $\mathcal{T}$ of n triples, the MAX3DM instance has optimal solution OPT(I) $= k^\star$ (i.e., a total of $k^\star$ triples in the solution set $\mathcal{S}$) iff the reduced MAXMAPF$_{\langle ms,B\leq 3\rangle}$ (or MAX-diMAPF$_{\langle ms,B\leq 3\rangle}$) instance I' has optimal solution of OPT(I') $= (m + k^\star)$ agents arriving their respective goal vertices in makespan 3.*

Proof. Let $I'(O, P, Q)$ be a solution to I', where O, P and Q, respectively, are the total number of dashed-boxes which receives two, one, and zero Layer A agents after Step 1. We require that $I'(O, P, Q) = 2\cdot O + 2\cdot P + Q$. That is, no collisions allowed, and in total $(P + Q)$ v-agents, and $(2\cdot O + P)$ Layer A agents, arrive at their respective goals in the solution. Nevertheless, for any of the "dashed-box" where there are two Layer A agents crossing it, we can always trade one of them for the v-agent initially in the same "dashed-box". This means, we always have the equivalence of $I'(O, P, Q) = I'(0, (O + P), Q)$. Consequently, any optimal solution subject to the condition of $O = 0$, is indeed a global optimal solution. Note also that regardless, $O+P+Q$ always equals m. Hence the task here is reduced to maximize the value of $(O + P)$, and we have that *OPT*(I) $= k^\star$ iff *OPT*(I') $= I'(0, k^\star, Q) = m + k^\star$. □

Proposition 2. *There exists an L-reduction from MAX3DM to MAXMAPF$_{\langle ms,B\leq 3\rangle}$ (or MAX-diMAPF$_{\langle ms,B\leq 3\rangle}$).*

Proof. We have constructed an L-reduction, with parameters $a = \frac{7}{3}$ and $b = 1$: 1) For each instance I of Π we can compute in polytime an instance I' of Π'. 2) Since for any *MAX3DM* there exists a polytime algorithm that matches $\frac{3}{4}$ triples (Theorem 3), we know that $k^\star \geq \frac{3}{4}\cdot m$, i.e., $m \leq \frac{4}{3}k^\star$. Hence *OPT*($I'$) $= (m + k^\star) \leq \frac{4}{3}k^\star + k^\star < \frac{7}{3}k^\star = \frac{7}{3}$*OPT*($I$) $= a\cdot$ *OPT*(I). 3) Given a solution of value $V' = (m + \tilde{k})$ to I', we can compute in polytime a solution of value $V = \tilde{k}$ to I such that $(k^\star - \tilde{k}) =$ *OPT*(I) $- V \leq$ *OPT*(I') $- V' = (m + k^\star) - (m + \tilde{k}) = b\cdot(k^\star - \tilde{k})$, where $b = 1$. □

Theorem 9. *There does not exist a $\frac{662}{665}$-approximation algorithm for MAXMAPF$_{\langle ms,B<3\rangle}$, or MAX-diMAPF$_{\langle ms,B<3\rangle}$, unless P = NP.*

Proof. Note that we get the number $\frac{662}{665}$ from the fact that $(\frac{7}{3})\cdot(\frac{662}{665}) - \frac{4}{3} = \frac{94}{95}$. Specifically, since we have an L-reduction here, we know that for any solution of value V' to I', which is an instance of the constructed *MAXMAPF*$_{\langle ms,B<3\rangle}$, or *MAX-diMAPF*$_{\langle ms,B<3\rangle}$, a solution of value V to I, which is an instance of *MAX3DM*, we have that *OPT*(I) $- V \leq$ (*OPT*(I') $- V'$). That is, $V \geq$ *OPT*(I) $-$ (*OPT*(I') $- V'$) $\geq$ *OPT*(I) $- (1-\alpha)\cdot$ *OPT*(I') $\geq$ *OPT*(I) $- (1-\alpha)\cdot\frac{7}{3}\cdot$ *OPT*(I) $\geq$ *OPT*(I) $\cdot\left(\frac{7}{3}\alpha - \frac{4}{3}\right)$. That is, if $\alpha > \frac{662}{665}$, $(\frac{7}{3}\alpha - \frac{4}{3})$ is greater than $\frac{94}{95}$. However, there does not exist a better than $\frac{94}{95}$-approximation algorithm for *MAX3DM* (Theorem 4). Hence α can not be greater than $\frac{662}{665} \approx 0.995$, unless *P* = *NP*. □

4 Inapproximability of *MAX-diMAPF*

In this section, we examine the complexity of *MAX-diMAPF*, i.e., the optimal variant of *diMAPF*, on maximizing the total number of agents reaching their goals. Specifically, we prove for the problem the non-approximability of any polytime algorithm within a certain factor. We prove a first result, and next the result is refined to a much tighter bound. We begin with constructing an L-reduction from *MAXE3SAT* to *MAX-diMAPF*.

Reduction We use Figure 5 to illustrate our reduction from *MAXE3SAT* to *MAX-diMAPF*. The reduction produces an acyclic graph.

We assume a *MAXE3SAT* instance with m clauses and n variables ($m = 5$, and $n = 4$ in the figure). Each variable is associated with an m-long mutex as pictured on Figure 6. This gadget includes four vertices v^i_{left}, v^i_{right}, v^i_{top}, and v^i_{btm}. We add edges (v^i_{left}, v^i_{top}) and (v^i_{left}, v^i_{btm}). Furthermore, there is an $m-1$-long path from v^i_{top} to v^i_{right}, and similarly from v^i_{btm} to v^i_{right}. Finally, the gadget includes a head of $m-1$ vertices that lead to v^i_{left}. The head of the gadget (including v^i_{left}) is populated with m r-agents. We also note that there are four connections to the rest of the graph: three inputs in, v, and $\neg v$ respectively at the start of the head, at v^i_{top}, and at v^i_{btm}, and one output out at v^i_{right}.

In Figure 5, the m-long mutexes are represented with dashed-boxes. They are connected sequentially so that the exit of one leads to the entrance of the next. The exit of this

[3]Given Theorem 4, the inapproximability theorem for *MAX3DM*, we know that it is sufficient to consider *MAX3DM* where each element has exactly two occurrences in the triples. Accordingly, in the proof for L-reduction, each dashed-box contains exactly one gadget (Figure 4).

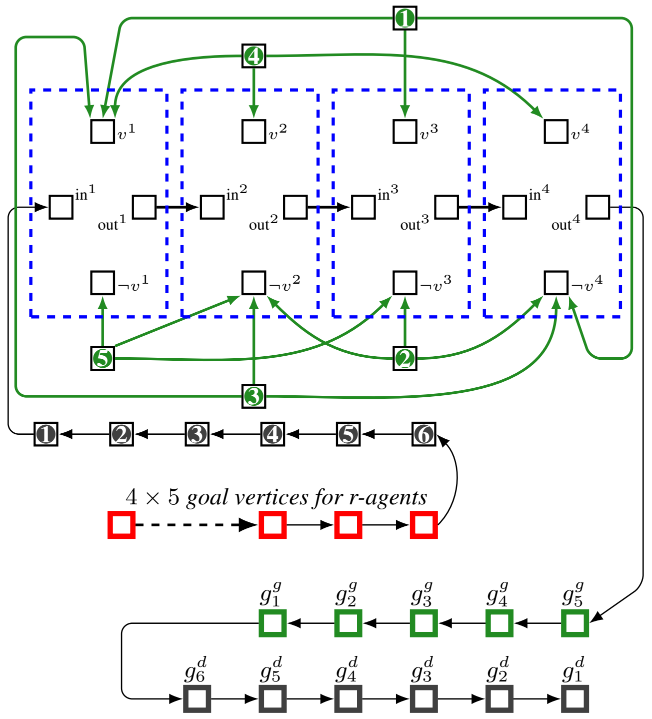

Figure 5: The *MAX-diMAPF* instance, which is constructed from the *MAXE3SAT* instance with $m = 5$ clauses and $n = 4$ variables : $(x_1 \vee x_3 \vee \neg x_4) \wedge (\neg x_2 \vee \neg x_3 \vee \neg x_4) \wedge (x_1 \vee \neg x_2 \vee \neg x_4) \wedge (x_1 \vee x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$. A 5-long mutex is represented by a dashed-box. An inside look of a dashed-box is presented in Figure 6. Four dashed-boxes ($n = 4$) are connected sequentially, with an input of chained vertices, and an output of chained vertices. For each one of the five clauses ($m = 5$), a vertex, which is the starting location of a g-agent, is created. Such a vertex (say j) is connected to v^i for each positive literal i in clause c_j, and to $\neg v^i$ for each negative literal $\neg i$ in the clause.

sequence leads to m vertices that are the goals of m g-agents followed by $m+1$ vertices that are the goal of d-agents. The input of the series of mutexes is connected to $m + 1$ vertices where the d-agents start and $n \times m$ vertices that are the goals of the r-agents. Finally, for each clause c_j, a vertex is created that is the starting location of a g-agent; this vertex is connected to v^i for each positive literal i in clause c_j, and to $\neg v^i$ for each negative literal $\neg i$ in the clause.

Each mutex contains $3m+1$ vertices. Totally $(3m+1) \cdot n$ vertices are introduced. Before mutexes, $(m + 1) + m \times n$ vertices are introduced. After mutexes, $2m + 1$ vertices are introduced. For each of the m clauses, a vertex is introduced. Totally, we have $3mn + n + m + 1 + mn + 2m + 1 + m = 4mn + 4m + n + 1 \in O(mn)$ vertices created. Hence the total number of agents created is also in $O(mn)$, and the total number of edges created is in $O(m^2 n^2)$. The reduction is in polytime.

Properties of the reduction: optimal solution of the reduced instance We first note that the r-agents cannot reach

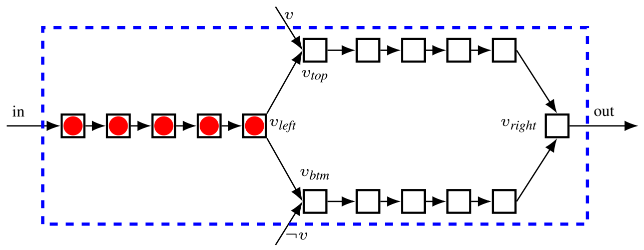

Figure 6: A 5-long mutex. This gadget is built such that, assuming the r-agents must move first but cannot escape the gadget, they will block one of the two paths $v_{top} \rightsquigarrow v_{right}$ and $v_{btm} \rightsquigarrow v_{right}$.

their goal location; there remains $2m + 1$ agents that might ($m + 1$ d-agents and m g-agents). We also note that, in the initial state, the r-agents block the paths through the graph for the d- and the g-agents. In order to let other agents pass, they can do one of following options:

1. For some variable i, the r-agents stay on the starting locations, blocking the path for the d-agents. This will lead to trivially sub-optimal solution (as shown below).
2. For each variable i, all r-agents move to the top or the bottom vertices of the gadget, and the r-agents of a gadget never all exit their starting gadget. Doing so opens exactly one of the two paths of the gadget (so that g-agents can go through either v^i_{top} or v^i_{btm} but not both).
3. For some variable i, all r-agents leave the gadget but they never reach the tail of the graph (they remain in some other gadget). This will free both v^i_{top} and v^i_{btm}, but will block both paths of some subsequent gadget (because there are only exactly enough vertices for the r-agents to leave a path open), meaning that the d-agents and g-agents that have not passed yet will be blocked.
4. For some variable i, all r-agents leave the gadget and some r-agents will eventually reach the tail. In order to open a new path, at least m r-agents should enter the tail. This will reduce the total number of agents reaching their goal destinations to at most $m+1$, which will be trivially below optimal.

We conclude from this that only the second option will be part of an optimal solution.

It is very simple to get the $m + 1$ d-agents to their goal: move the r-agents out of the way and get the d-agents through the graph.

Now consider the g-agents. We already know that the optimal solution requires moving the r-agents so that all d-agents can pass, and that this move will block either v^i_{top} or v^i_{btm}. This means that for g-agent j to reach its destination, it will need to have one of its v^i_{top}/v^i_{btm} open (assuming in the original instance, i is a variable, and c_j is a clause, either $i \in c_j$ or $\neg i \in c_j$), i.e., any *diMAPF* solution that sends the d-agents to their goal location is such that the g-agents form a satisfiable set of clauses. In other words, the optimal *diMAPF* solution with value $m+1+k$ can be converted into a solution to the original *MAXE3SAT* instance with value k.

From *MAXE3SAT* to the reduced problem Conversely, given a solution of value k to a *MAXE3SAT* instance, it is possible to build a solution to our reduced instance of value $m+1+k$ by moving the r-agents in the appropriate branch, having the d-agents reach their goal, and moving the g-agents for the satisfiable subset of clauses to their goal.

Proposition 3. *Let I be a MAXE3SAT instance with m clauses and n variables. The MAXE3SAT instance has optimal solution $OPT(I) = k^\star$ (i.e., a total of $k^\star$ clauses would be satisfied) iff the reduced MAX-diMAPF instance I' has optimal solution of $OPT(I') = (m+1+k^\star)$ agents arriving their respective goal vertices.*

Proof. This is a direct consequence of our discussion of the properties of the reduction. □

Proposition 4. *There exists an L-reduction from MAXE3SAT to MAX-diMAPF.*

Proof. We have constructed an L-reduction, with parameters $a = \frac{15}{7}$ and $b = 1$: 1) For each instance I of Π we can compute in polytime an instance I' of Π'. 2) Since for any *MAXE3SAT* there exists an algorithm that satisfies at least $\frac{7}{8}$ of the clauses (Theorem 2), we know that $k^\star \geq \frac{7}{8} \cdot m$, i.e., $m \leq \frac{8}{7}k^\star$. Hence $OPT(I') = (m+1+k^\star) \leq \frac{8}{7}k^\star + k^\star + 1 = (\frac{15}{7} + \frac{1}{k^\star}) \cdot k^\star$. Asymptotically, as the input problem size increases, $\frac{1}{k^\star} \to 0$, and the RHS approaches $\frac{15}{7}k^\star$. Hence we further have $OPT(I') \leq \frac{15}{7}k^\star = \frac{15}{7}OPT(I) = a \cdot OPT(I)$. 3) Given a solution of value $V' = (m+1+\tilde{k})$ to I', we can compute in polytime a solution of value $V = \tilde{k}$ to I such that $(k^\star - \tilde{k}) = OPT(I) - V \leq OPT(I') - V' = (m+1-k^\star) - (m+1-\tilde{k}) = b \cdot (k^\star - \tilde{k})$, where $b = 1$. □

Theorem 10. *There does not exist an α-approximation algorithm for the MAX-diMAPF problem for constant $\alpha > \frac{113}{120} \approx 0.942$, unless P = NP.*

Proof. Note that we get the number $\frac{113}{120}$ from the fact that $(\frac{15}{7})(\frac{113}{120}) - \frac{8}{7} = \frac{7}{8}$. Specifically, since we have an L-reduction here, we know that for any solution of value V' to I', which is an instance of the constructed *MAX-diMAPF*, a solution of value V to I, which is an instance of *MAXE3SAT*, we have $OPT(I) - V \leq (OPT(I') - V')$. That is, $V \geq OPT(I) - (OPT(I') - V') \geq OPT(I) - (1-\alpha) \cdot OPT(I')$ $\geq OPT(I) - (1-\alpha) \cdot \frac{15}{7} \cdot OPT(I) \geq OPT(I) \cdot \left(\frac{15}{7}\alpha - \frac{8}{7}\right)$.

Hence if $\alpha > \frac{113}{120}$, $(\frac{15}{7}\alpha - \frac{8}{7})$ is greater than $\frac{7}{8}$. However there does not exist a better than $\frac{7}{8}$-approximation algorithm for *MAXE3SAT* (Theorem 2). Hence α can not be greater than $\frac{113}{120} \approx 0.942$, unless *P* = *NP*. □

From the fact of $k^\star \geq \frac{7}{8}$ for *MAXE3SAT* (Theorem 2), we can get a stronger result.

Corollary 11. *There does not exist an α-approximation algorithm for the MAX-diMAPF problem for constant $\alpha > \frac{57}{64} \approx 0.891$, unless P = NP.*

Proof. We can reduce the total number of dark-agents from $(m+1)$ down to $(\lceil \frac{m}{8} \rceil + 1)$. We need to make sure that the optimal solution is not simply the m green-agents arriving their goals and the dark-agents staying put. However, we already know that $k^\star \geq \frac{7}{8} \cdot m$, hence $(\lceil \frac{m}{8} \rceil + 1)$ dark-agents only, are needed in order to guarantee that the dark-agents are involved in the optimal solution. Hence $OPT(I')$ is refined to $(\lceil \frac{m}{8} \rceil + 1 + k^\star)$, which is asymptotically less than $\frac{8}{7} \cdot k^\star$. That is, now we have $a = \frac{8}{7}$. As a consequence, we get that $\alpha > \frac{57}{64} \approx 0.891$ (arithmetical steps skipped). □

5 Conclusions

This research investigates the computational approximability of several optimal *MAPF* problems. It demonstrates that many of these problems, in their general form, are inapproximable in polytime, unless *P* = *NP*. Specifically:

- Optimal *MAPF* on both directed graphs (Theorem 5) and undirected graphs (Corollary 7)[4], with a makespan ≤ 3, are NP-complete.
- Optimal *MAPF* constrained on makespan, on both directed and undirected graphs, are not α-approximable in polytime for $\alpha \leq \frac{4}{3}$, unless *P* = *NP* (Corollaries 6 and 7).
- If the agents, and their respective goals, are categorized into two types only, α can be reduced to $\frac{3}{2}$ (Corollary 8), which is actually the extreme case NP-hardness result, as the makespan bound is only 2. The general version of this problem in the literature is called Colored *MAPF* (Solovey and Halperin 2014; Ma and Koenig 2016), where agents are grouped into teams. There are only two teams in our result.
- $\mathit{MAXMAPF}_{\langle ms,B\leq 3\rangle} \big/ \mathit{MAX\text{-}diMAPF}_{\langle ms,B\leq 3\rangle}$, an optimal variant of *MAPF*/*diMAPF* with makespan 3, that also aims to maximize the number of agents reaching their goals, does not have an approximation algorithm with a ratio greater than 0.995, unless *P* = *NP* (Theorem 9).
- The problem *MAX-diMAPF*, an optimal variant of *diMAPF* that aims to maximize the number of agents that reach their goals, does not have an approximation algorithm with a ratio greater than 0.891, unless *P* = *NP* (Theorem 10, Corollary 11).

To recap, among several other results, we have provided a first set of inapproximability results for both *MAPF* and *diMAPF* problems when maximizing the total number of agents that can reach their goal vertices (Theorems 9/10).

Future research includes investigating ways to tighten the inapproximability bounds established in the paper. It would also be interesting to check if these results can be extended to *MAPF* on well-studied restricted graphs (Fioravantes et al. 2024), such as grids, planar graphs, trees, etc. Meanwhile, from the perspective of parameterized complexity, we are interested in identifying parameters that might also make some of these problems tractable (Eiben, Ganian, and Kanj 2023; Deligkas et al. 2024).

[4]Corollary 7 was discovered earlier (Ma et al. 2016).

Acknowledgments

The authors thank all the reviewers for their valuable feedback. Xing Tan gratefully acknowledges the Discovery Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), and the research grant from the Faculty of Science & Environmental Studies, Lakehead University.

References

Banfi, J.; Basilico, N.; and Amigoni, F. 2017. Intractability of time-optimal multirobot path planning on 2D grid graphs with holes. *IEEE Robotics and Automation Letters*, 2(4): 1941–1947.

Barer, M.; Sharon, G.; Stern, R.; and Felner, A. 2014. Sub-optimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In *Proc. of the 21st ECAI*, 961–962.

Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Tolpin, D.; Betzalel, O.; and Shimony, E. 2015. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding. In *Proc. of the 25th IJCAI*, 740–746.

Chlebík, M.; and Chlebíková, J. 2006. Complexity of approximating bounded variants of optimization problems. *Theor. Comput. Sci.*, 354(3): 320–338.

Cohen, L.; Uras, T.; Kumar, T. K. S.; Xu, H.; Ayanian, N.; and Koenig, S. 2016. Improved solvers for bounded-suboptimal multi-agent path finding. In *Proc. of the 25th IJCAI*, 3067–3074.

Cygan, M. 2013. Improved approximation for 3-Dimensional matching via bounded pathwidth local search. In *Proc. of the 54th FOCS*, 509–518.

de Wilde, B.; ter Mors, A. W.; and Witteveen, C. 2013. Push and rotate: cooperative multi-agent path planning. In *Proc. of the 12th AAMAS*, 87–94.

Deligkas, A.; Eiben, E.; Ganian, R.; Kanj, I.; and Ramanujan, M. S. 2024. Parameterized algorithms for coordinated motion planning: Minimizing energy. In *Proc. of the 51st ICALP*, 53:1–53:18.

Eiben, E.; Ganian, R.; and Kanj, I. 2023. The parameterized complexity of coordinated motion planning. In *Proc. of the 39th SoCG*, 28:1–28:16.

Fioravantes, F.; Knop, D.; Kristan, J. M.; Melissinos, N.; and Opler, M. 2024. Exact algorithms and lowerbounds for multiagent path finding: Power of treelike topology. In *Proc. of the 38th AAAI*, 17380–17388.

Garey, M.; and Johnson, D. 1979. *Computers and intractability - a guide to NP-completeness*. Cambridge, MA, USA: San Francisco: W.H. Freeman and Company.

Geft, T.; and Halperin, D. 2022. Refined hardness of distance-optimal multi-agent path finding. In *Proc. of the 21st AAMAS*, 481–488.

Håstad, J. 2001. Some optimal inapproximability results. *J. ACM*, 48(4): 798–859.

Johnson, D. S. 1974. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3): 256–278.

Karp, R. M. 1972. Reducibility among combinatorial problems. In Miller, R. E.; Thatcher, J. W.; and Bohlinger, J. D., eds., *Complexity of Computer Computations*, 85–103.

Kornhauser, D.; Miller, G.; and Spirakis, P. 1984. Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In *Proc. of the 25th FOCS*, 241–250.

Ma, H.; and Koenig, S. 2016. Optimal target assignment and path finding for teams of agents. In *Proc. of the 15th AAMAS*, 1144–1152.

Ma, H.; Tovey, C. A.; Sharon, G.; Kumar, T. K. S.; and Koenig, S. 2016. Multi-agent path finding with payload transfers and the package-exchange robot-routing problem. In *Proc. of the 30th AAAI*, 3166–3173.

Ma, H.; Wagner, G.; Felner, A.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2018. Multi-agent path finding with deadlines. In *Proc. of the 27th IJCAI*, 417–423.

Nebel, B. 2020. On the computational complexity of multi-agent pathfinding on directed graphs. In *Proc. of the 30th ICAPS*, 212–216.

Nebel, B. 2023. The small solution hypothesis for MAPF on strongly connected directed graphs is true. In *Proc. of the 33rd ICAPS*, 304–313.

Nebel, B. 2024. The computational complexity of multi-agent pathfinding on directed graphs. *Artificial Intelligence*, 328: 104063.

Okumura, K.; Bonnet, F.; Tamura, Y.; and Défago, X. 2022. Offline time-independent multi-agent path planning. In *Proc. of the 31st IJCAI*, 4649–4656.

Papadimitriou, C. H.; and Yannakakis, M. 1991. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3): 425–440.

Salzman, O.; and Halperin, D. 2016. Asymptotically near-optimal RRT for fast, high-quality motion planning. *IEEE Trans. on Robotics*, 32(3): 473–483.

Salzman, O.; and Stern, R. 2020. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems. In *Proc. of the 19th AAMAS*, 1711–1715.

Šišlák, D.; Volf, P.; and Pěchouček, M. 2010. Agent-based cooperative decentralized airplane-collision avoidance. *IEEE Trans. on Intelligent Transportation Systems*, 12(1): 36–46.

Solovey, K.; and Halperin, D. 2014. k-color multi-robot motion planning. *The International Journal of Robotics Research*, 33(1): 82–97.

Standley, T. 2010. Finding optimal solutions to cooperative pathfinding problems. In *Proc. of the 24th AAAI*, 173–178.

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 *Proc. of the 12th SoCS, 2019*, 151–159.

Surynek, P. 2010. An optimization variant of multi-robot path planning is intractable. In *Proc. of the 24th AAAI*, 1261–1263.

Tan, X.; and Bercher, P. 2023. Intractability of optimal multi-agent pathfinding on directed graphs. In *Proc. of the 26th ECAI*, 2315–2321.

Vodrázka, J.; Barták, R.; and Svancara, J. 2020. On modelling multi-agent path finding as a classical planning problem. In *Proc. of the 32nd ICTAI*, 23–28.

Wurman, P. R.; D'Andrea, R.; and Mountz, M. 2008. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine*, 29(1): 9–20.

Yannakakis, M. 1994. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17(3): 475–502.

Yu, J. 2016. Intractability of optimal multirobot path planning on planar graphs. *IEEE Robotics and Automation Letters*, 1(1): 33–40.

Yu, J.; and LaValle, S. M. 2013. Structure and intractability of optimal multi-robot path planning on graphs. In *Proc. of the 27th AAAI*, 1443–1449.