A Competitive Analysis of Online Multi-Agent Path Finding*

Hang Ma
Simon Fraser University
hangma@sfu.ca

Abstract

We study online Multi-Agent Path Finding (MAPF), where new agents are constantly revealed over time and all agents must find collision-free paths to their given goal locations. We generalize existing complexity results of (offline) MAPF to online MAPF. We classify online MAPF algorithms into different categories based on (1) controllability (the set of agents that they can plan paths for at each time) and (2) rationality (the quality of paths they plan) and study the relationships between them. We perform a competitive analysis for each category of online MAPF algorithms with respect to commonly-used objective functions. We show that a naive algorithm that routes newly-revealed agents one at a time in sequence achieves a competitive ratio that is asymptotically bounded from both below and above by the number of agents with respect to flowtime and makespan. We then show a counter-intuitive result that, if rerouting of previously-revealed agents is not allowed, any rational online MAPF algorithms, including ones that plan optimal paths for all newly-revealed agents, have the same asymptotic competitive ratio as the naive algorithm, even on 2D 4-neighbor grids. We also derive constant lower bounds on the competitive ratio of any rational online MAPF algorithms that allow rerouting. The results thus provide theoretical insights into the effectiveness of using MAPF algorithms in an online setting for the first time.

1 Introduction

Online Multi-Agent Path Finding (MAPF) (Švancara et al. 2019) models the problem of finding collision-free paths for a stream of incoming agents in a given region. Its applications include autonomous intersection management (Dresner and Stone 2008), UAV traffic management (Ho et al. 2019), video games (Ma et al. 2017b), and automated warehouse systems (Wurman, D'Andrea, and Mountz 2008). For example, Figure 1 shows the typical grid layout of part of a modern automated warehouse, where warehouse robots (orange squares) need to move inventory pods between their storage locations (green cells) and inventory stations (squares in purple and pink). The narrow corridors in the storage region are single-direction lanes where traffic is controlled by a traffic-rule-based system. However, paths are not known and must be planned for warehouse robots in the intersection region (red rectangle) that is often highly congested. Warehouse robots constantly enter this region at their given start cells and must plan collision-free paths to and exit at their given goal cells (along an edge of the rectangle). The problem in such applications is online because each agent is known only when it is about to enter such a region and the future arrivals of agents are not known ahead of time.

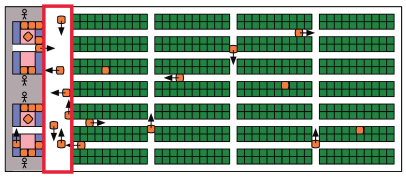


Figure 1: The 2D grid layout of part of an Amazon Robotics automated warehouse, reproduced from Wurman, D'Andrea, and Mountz (2008).

Existing research has conducted empirical evaluations of several online MAPF algorithms (Švancara et al. 2019) based on recent techniques for (offline) MAPF (Stern et al. 2019). However, there is still a lack of theoretical understanding of online MAPF and its algorithms. In this paper, we thus perform a theoretical analysis from the points of view of competitive analysis and complexity theory.

1.1 Related Work

Offline MAPF: Online MAPF is an extension of the well-studied problem of (offline) MAPF (Ma and Koenig 2017; Stern et al. 2019), where all agents are known and start routing at the same time. MAPF is NP-hard to solve optimally for flowtime (the sum of the arrival times of all agents at their goal locations) minimization and to approximate within any constant factor less than 4/3 for makespan (the maximum of the arrival times of all agents at their goal locations) minimization (Surynek 2010; Yu and LaValle 2013b; Ma et al. 2016). It is NP-hard to solve optimally even on planar graphs (Yu 2015) and 2D 4-neighbor grids (Banfi,

*This work was supported by the Natural Sciences and Engineering Research Council (NSERC) under grant number RGPIN-2020-06540.

This version of the paper is intended to update the version published in the Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICASP 2021). The definition of *makespan* (Section 2) has been corrected from $\max_{a_i \in \mathcal{A}}(t_i^{(g)} - r_i)$ to $\max_{a_i \in \mathcal{A}} t_i^{(g)}$. Other parts of the paper are not affected.

Basilico, and Amigoni 2017). MAPF algorithms include reductions to other combinatorial problems (Yu and LaValle 2013a; Erdem et al. 2013; Surynek et al. 2016) and specialized algorithms (Luna and Bekris 2011; Wang and Botea 2011; Sharon et al. 2013, 2015; Boyarski et al. 2015; Cohen et al. 2018; Ma et al. 2019a; Li et al. 2019a,b; Lam et al. 2019; Gange, Harabor, and Stuckey 2019; Li et al. 2020).

Online Problems: Ma et al. (2017a) and Ma et al. (2019b) have considered an online version of MAPF where a given set of agents must attend to a stream of tasks, consisting of (sub-)goal locations to be assigned to the agents, that appear at unknown times. This version considers the entire environment instead of a region of a system and thus does not consider the appearance and disappearance of agents. Švancara et al. (2019) and Ho et al. (2019) have considered another online version of MAPF, similar to the setting of this paper, where a stream of agents with preassigned goal location appear at unknown times. Algorithms for solving such online problems reduce each problem to a sequence of (offline) MAPF sub-problems that are solved by a MAPF algorithm. The effectiveness of these algorithms is characterized by objective functions that measure how soon the tasks are finished or the agents are routed to their goal locations. Existing study on online versions of MAPF has been empirical only. For example, both Ma et al. (2017a) and Švancara et al. (2019) have experimentally shown that algorithms that allow agents (that have paths already) to replan their paths and reroute tend to be more effective than those that do not. However, there is still a lack of theoretical understanding of solving MAPF in an online setting.

1.2 Assumptions and Contributions

We follow most of the notations of Švancara et al. (2019) and consider the setting where new agents can wait infinitely long before entering a given region and agents disappear upon exiting from the region because (1) existing online MAPF algorithms have been designed and tested only for this setting (Ho et al. 2019; Švancara et al. 2019), although other settings concerning what happens before agents enter the region and after agents leave the region have (only) been mentioned (briefly) by Stern et al. (2019); Švancara et al. (2019) and (2) queuing at entrances and exits of such an intersection region is handled by a task-level planner/scheduler with reserved queuing spaces (for example, queues in inventory stations and along the single-lane corridors in the storage region) in automated warehouses (Wurman, D'Andrea, and Mountz 2008; Kou et al. 2020) and many other real-world systems. We view the problem from the point of view of competitive analysis and thus assume that the algorithms have no knowledge of future arrivals of agents, as in the case of all existing online MAPF algorithms (Ho et al. 2019; Švancara et al. 2019), although such knowledge might be learned in practice.

As our first contribution, we formalize online MAPF as an extension of (offline) MAPF and demonstrate how to generalize existing NP-hardness and inapproximability results for MAPF to online MAPF.

As our second contribution, we classify online MAPF algorithms based on (1) different controllability assumptions, namely at what time the system can plan paths for which sets of agents, into three categories: PLAN-NEW-SINGLE that plans only a path for one newly-revealed agent at a time, PLAN-NEW that plans paths only for newly-revealed agents, and PLAN-ALL that plans paths for all known agents and thus allows rerouting and (2) different rationality, namely how effective the planned paths are: optimally-rational algorithms that plan optimal paths for the given set of agents and rational algorithms (which are, in our opinion, the only algorithms worth considering, assuming no knowledge of future arrivals of agents) that plan paths at least asymptotically as effective as the naive baseline algorithm SEQUENCE that routes newly-revealed agents one at a time in sequence. These classifications cover all existing online MAPF algorithms in Švancara et al. (2019) and different settings, for example, where rerouting of robots is always allowed (Ma et al. 2017a) or disallowed (Ho et al. 2019), in real-world systems. The relationships between these algorithms are summarized in Figure 2.

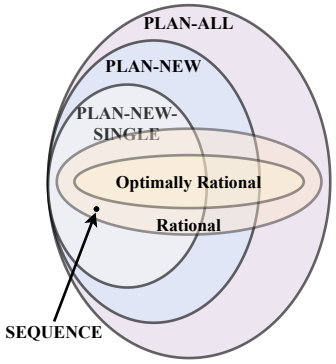


Figure 2: Relationships between online MAPF algorithms.

As our third contribution, we study online MAPF algorithms under the competitive analysis framework. Specifically, we demonstrate how an arbitrary online MAPF algorithm can be rationalized and show that the competitive ratios of all rational online MAPF algorithms with respect to flowtime and makespan are both bounded from above by $\mathcal{O}(m)$ for an input sequence of m agents. We then show that (1) the bounds are tight for all rational algorithms in PLAN-NEW-SINGLE and PLAN-NEW, (2) the competitive ratio is at least 4/3 with respect to flowtime and 3/2 with respect to makespan for all rational algorithms in PLAN-ALL, and (3) the competitive ratio is infinite with respect to latency for all rational algorithms. The results hold even for optimally-rational algorithms and on 4-neighbor 2D grids. Therefore, for the first time, we provide theoretical insights into the effectiveness of using MAPF algorithms in an online setting (Salzman and Stern 2020) and address some of the long-standing open questions such as whether planning for multiple agents is more effective than planning for only one agent at a time in an online setting, whether algorithms that allow rerouting are more effective than those that disallow, and whether acting optimally rationally can improve the effectiveness. The results are summarized in Table 1.

This version of the paper is intended to update the version published in the Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICASP 2021). The definition of *makespan* (Section 2) has been corrected from $\max_{a_i \in \mathcal{A}}(t_i^{(g)} - r_i)$ to $\max_{a_i \in \mathcal{A}} t_i^{(g)}$. Other parts of the paper are not affected.

Controllability		PLAN-NEW-SINGLE			PLAN-NEW		PLAN-ALL	
Objective Function	Competitive Ratio Bounds	SEQUENCE	Rational	Optimally Rational	Rational	Optimally Rational	Rational	Optimally Rational
flowtime	upper	$\mathcal{O}(m)$ [Thm. 7]	$\mathcal{O}(m)$ [Thm. 11]					
	lower (even on 2D grids)	$\Omega(m)$ [Thm. 13]					4/3 [Thm. 18]	
makespan	upper	$\mathcal{O}(m)$ [Thm. 8]	$\mathcal{O}(m)$ [Thm. 12]					
	lower (even on 2D grids)	$\Omega(m)$ [Thm. 14]					3/2 [Thm. 19]	
latency	(even on 2D grids)	∞ [Obs. 20]						

Table 1: Summary of main competitiveness results.

2 Online Multi-Agent Path Finding

In an online MAPF instance, we are given a connected undirected graph $G = (V, E)$, whose vertices V represent locations and whose edges E represent connections between locations that the agents can traverse. We consider a finite input sequence of m agents $a_1, a_2, \ldots, a_m$. Each agent a_i is characterized by a *start vertex* $s_i \in V$, a *goal vertex* $g_i \in V$ that is different from the start vertex, and a non-negative integer *release time* r_i, at which the agent is *revealed* (appears and is added to the system) and available for routing. Release times are not known to online MAPF algorithms. Without loss of generality, we assume that the agents are given in non-decreasing order of its release time, that is, $r_1 \leq r_2 \leq \ldots \leq r_m$. Each agent a_i can choose to start at any integer *starting time* $t_i^{(s)} \geq r_i$, where it is added to graph G at its start vertex. At each discrete time step, each agent a_i either moves to an adjacent vertex or waits at the same vertex when it is in graph G. When the agent arrives at its goal vertex at *arrival time* $t_i^{(g)}$, it is removed from graph G (upon its arrival at time step $t_i^{(g)}$).

Let $\pi_i(t) \in V$ denote the vertex of agent a_i at time step t. A *path* $\pi_i = \langle \pi_i(t_i^{(s)}), \pi_i(t_i^{(s)}+1), \ldots, \pi_i(t^{(g)}) \rangle$ for agent a_i satisfies the following condition: (1) The agent starts at its start vertex at the starting time $t_i^{(s)}$, that is, $\pi_i(t_i^{(s)}) = s_i$. (2) The agent ends at its goal vertex at the arrival time $t_i^{(g)}$, that is, $\pi_i(t_i^{(g)}) = g_i$ (but is removed from the graph upon its arrival and does not collide with other agents at time step $t_i^{(g)}$). (3) The agent always either moves to an adjacent vertex or waits at the same vertex between two consecutive time steps when in graph G, that is, for all time steps $t = t^{(s)}, \ldots, t_i^{(g)}$, $(\pi_i(t), \pi_i(t+1)) \in E$ or $\pi_i(t+1) = \pi_i(t)$. Path π is implicitly augmented with a null vertex $\perp \notin V$ for all time steps $t \in [0, \infty) \setminus [t_i^{(s)}, t_i^{(g)}]$. Every two agents should avoid collisions with each other (only) when they are both in graph G: A *vertex collision* occurs if two agents occupy the same vertex (except for their goal vertices) at time step t. An *edge collision* occurs if two agents traverse the same edge in opposite directions between time steps t and $t+1$.

A *plan* for a set $\mathcal{A}$ of agents consists of a path π_i assigned to each agent $a_i \in \mathcal{A}$. Let $dist_i$ denote the length of the shortest path (optimal path cost) from vertex s_i to vertex g_i in graph G. The *service time* $t_i^{(g)} - r_i$ of agent a_i is the number of time steps for the agent to arrive at its goal vertex since it is revealed. We consider three common objective functions:

1. The *flowtime* $\sum_{a_i \in \mathcal{A}} (t_i^{(g)} - r_i)$ is the sum of the service times $t_i^{(g)} - r_i$ of all agents in $\mathcal{A}$.
2. The *makespan* $\max_{a_i \in \mathcal{A}} t_i^{(g)}$ is the maximum of the arrival times of all agents in $\mathcal{A}$.[1]
3. The *latency* $\sum_{a_i \in \mathcal{A}} (t_i^{(g)} - r_i - dist_i)$ is the sum of the differences between the service times $t_i^{(g)} - r_i$ and the distances $dist_i$ of all agents in $\mathcal{A}$.[2]

Trivially, an online MAPF plan minimizes the flowtime if and only if it minimizes the latency, which can be rewritten as $\sum_{a_i \in \mathcal{A}} (t_i^{(g)} - r_i) - \sum_{a_i \in \mathcal{A}} dist_i$.

An online MAPF *solution* is a plan for all agents $a_1, \ldots, a_m$ whose paths are collision-free.

2.1 Computational Complexity

We show that online MAPF is NP-hard to solve (bounded-sub)optimally in general even with offline algorithms that know all agents a priori. Similar to Ma et al. (2016, 2018), we use a reduction from an NP-complete version of the Boolean satifiability problem, called $\leq$3,=3-SAT (Tovey 1984). A $\leq$3,=3-SAT instance consists of N Boolean variables and M disjunctive clauses where each variable appears in exactly three clauses, uncomplemented at least once and complemented at least once, and each clause contains at most three literals. Its decision question asks whether there exists a satisfying assignment.

Theorem 1. *For any $\epsilon > 0$, it is NP-hard to find a $4/3 - \epsilon$-approximate solution to online MAPF for makespan minimization, even if all agents are known a priori.*

The complete proof is given in the appendix. In the proof of Theorem 1, the online MAPF instance reduced from the given $\leq$3,=3-SAT instance has the property that the length of every path from the start vertex to the goal vertex of every agent is at least three. Therefore, if the makespan is three, then every agent arrives at its goal vertex in exactly three time steps and the flowtime is $3m$. Moreover, if the makespan exceeds three, then the flowtime exceeds $3m$, yielding the following corollary.

Corollary 2. *It is NP-hard to find an optimal solution to online MAPF for flowtime minimization, even if all agents are known a priori.*

In the proof of Theorem 1, the constructed online MAPF instance has a solution with zero latency if and only if the

[1] Švancara et al. (2019) claims that the makespan measure is problematic (probably because an infinite sequence of agents was considered there) and does not consider it. However, it is regarded as an important measure in the literature (Surynek 2010; Yu and LaValle 2013a), which corresponds to the earliest time when all (finitely many) transportation requests are served in practice.

[2] The latency measure has been used in the multi-vehicle transportation research, for example, in the context of online ride and delivery services (Das et al. 2018), and was first proposed by Švancara et al. (2019) for online MAPF.

This version of the paper is intended to update the version published in the Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICASP 2021). The definition of *makespan* (Section 2) has been corrected from $\max_{a_i \in \mathcal{A}}(t_i^{(g)} - r_i)$ to $\max_{a_i \in \mathcal{A}} t_i^{(g)}$. Other parts of the paper are not affected.

given ≤3,=3-SAT instance is satisfiable. Any algorithm for online MAPF with any constant approximation ratio c thus computes a solution with zero latency whenever the given ≤3,=3-SAT instance is satisfiable and thus solves ≤3,=3-SAT, yielding the following corollary.

Corollary 3. *For any $c > 0$, it is NP-hard to find a c-approximate solution to online MAPF for latency minimization, even if all agents are known a priori.*

3 Online MAPF Algorithms

Online MAPF algorithms use the following assumption: For any given MAPF instance, the agents are partitioned into K disjoint sets $\mathcal{A}_1, \ldots, \mathcal{A}_K$ based on their release times (a total of K different values), where agents in each set $\mathcal{A}_k$ have the same release time $r_{\mathcal{A}_k}$ for all $k \in [K]$ (where $[K] = \{1, \ldots, K\}$) and the sets $\mathcal{A}_k$ are indexed in increasing order of $r_{\mathcal{A}_k}$. Let $\mathcal{A}_{\leq k} = \bigcup_{k' \in [k]} \mathcal{A}_{k'}$ denote the set of all revealed agents by release time $r_{\mathcal{A}_k}$, for all $k \in [K]$, and $\mathcal{A}_{<k} = \bigcup_{k' \in [k-1]} \mathcal{A}_{k'}$ the set of all previously-revealed agents at each release time $r_{\mathcal{A}_k}$, for all $k \in [2, K]$. At each $r_{\mathcal{A}_k}$, an online MAPF algorithm calls an offline path-finding algorithm to plan paths so that all revealed agents have paths.

3.1 Controllability

We categorize online MAPF algorithms based on different *controllability assumptions* about for which *controllable set* $\mathcal{A}^{\mathcal{C}}$ of agents they can plan paths at each time when they call an offline path-finding algorithm. Algorithm 1 shows the pseudo-code of a template of online MAPF algorithms. At each release time $r_{\mathcal{A}_k}$, an online MAPF algorithm considers each controllable set of agents [Line 1]. It then treats all agents that have an already planned path and are not in the controllable set as *dynamic obstacles* that follow their already planned paths [Line 2] and computes a plan for all agents in the controllable set [Line 3]. We highlight the following three controllability categories.

1. [$\mathcal{A}^{\mathcal{C}} = \{a_i\}, \forall a_i \in \mathcal{A}_k$] Algorithms in **PLAN-NEW-SINGLE** call an offline single-agent path-finding algorithm to plan a path for one (newly-revealed) agent a_i in $\mathcal{A}_k$ at a time in increasing order of the indices of the agents and treat all agents a_j with $j < i$ as dynamic obstacles, until all agents in $\mathcal{A}_k$ have paths.
2. [$\mathcal{A}^{\mathcal{C}} = \mathcal{A}_k$] Algorithms in **PLAN-NEW** call an offline MAPF algorithm to plan paths for all newly-revealed agents, namely agents in $\mathcal{A}_k$, and treat all previously-revealed agents as dynamic obstacles.
3. [$\mathcal{A}^{\mathcal{C}} = \mathcal{A}_{\leq k}$] Algorithms in **PLAN-ALL** call an offline MAPF algorithm to plan paths for all revealed agents from time step $r_{\mathcal{A}_k}$ on, thus allowing previously-revealed agents to change their paths from time step $r_{\mathcal{A}_k}$ on.

Therefore, online MAPF algorithms in the same category differ from each other only in the offline single/multi-agent path-finding algorithm they used to solve each single/multi-agent path-finding problem. Trivially, PLAN-NEW-SINGLE is a subset of PLAN-NEW, which, in turn, is a subset of PLAN-ALL, since any online MAPF algorithm in PLAN-NEW-SINGLE can be viewed as a special case of PLAN-NEW that plans a path for each newly-revealed agent in sequence and any online MAPF algorithm in PLAN-NEW can be viewed as a special case of PLAN-ALL that always plans the same paths as the already planned ones for all previously-revealed agents.

Algorithm 1: Online MAPF Algorithm Template

Input: online MAPF instance
/ system executes at release time $r_{\mathcal{A}_k}$ */*
1 **foreach** controllable set $\mathcal{A}^{\mathcal{C}}$ **do**
2 $\quad \bar{\mathcal{A}} \leftarrow \{a_i | a_i$ has a path **and** $a_i \notin A^{\mathcal{C}}\}$;
3 $\quad$ Compute a plan for $\mathcal{A}^{\mathcal{C}}$ that treats $\bar{\mathcal{A}}$ as dynamic obstacles;
/ system advances to the next release time */*

In practice, different controllability assumptions correspond to different types of real-world systems: Algorithms in PLAN-NEW-SINGLE can easily be adapted to most real-time distributed systems since agents make decisions individually and require less communication with each other. PLAN-NEW corresponds to centralized systems where agents cannot be rerouted easily, for example, automated parcel sortation centers with agents moving at high speeds (Kou et al. 2020). PLAN-ALL corresponds to centralized systems that allow frequent rerouting, for example, a recent proposal of automated train (re-)scheduling systems by the Swiss Federal Railways (Mohanty et al. 2020; Li et al. 2021). Controllability often depends on specific applications and might also be viewed as part of the problem definition.

3.2 Optimal Rationality

We now discuss one category of online MAPF algorithms based on the quality of paths the algorithms plan for the controllable set of agents at each time. This categorization is orthogonal to the controllability assumptions and more algorithmic than application-specific.

Recent research (Švancara et al. 2019) has shown that "snapshot-optimal" online MAPF algorithms in PLAN-ALL that (use an optimal offline MAPF algorithm to) compute a plan for all revealed agents with the smallest flowtime at each release time tend to result in an online MAPF solution with small flowtime. We generalize this notion of optimality to algorithms in all the above three controllability categories.

Definition 4 (Optimal Rationality)**.** A plan for a given controllable set $\mathcal{A}^{\mathcal{C}}$ of agents at a given release time is *optimally-rational* if and only if it results in a plan for all planned agents (agents that have paths) with the smallest cost with respect to a given objective function (under a given controllability assumption). An online MAPF algorithm is *optimally-rational* if and only if it computes an optimally-rational plan for the controllable set of agents at each time when it calls an offline path-finding algorithm.

Trivially, an online MAPF algorithm is optimally-rational with respect to flowtime if and only if it is optimally-rational with respect to latency. We now give examples of offline path-finding algorithms that can be used in optimally-rational online MAPF algorithms under different controllability assumptions with respect to flowtime and makespan.

This version of the paper is intended to update the version published in the Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICASP 2021). The definition of *makespan* (Section 2) has been corrected from $\max_{a_i \in \mathcal{A}}(t_i^{(g)} - r_i)$ to $\max_{a_i \in \mathcal{A}} t_i^{(g)}$. Other parts of the paper are not affected.

Examples of Optimally-Rational Algorithms: Optimally-rational algorithms in PLAN-NEW-SINGLE with respect to flowtime and makespan can both call Space-Time A* (Silver 2005) or Safe Interval Path Planning (SIPP) (Phillips and Likhachev 2011) at release time $r_{\mathcal{A}_k}$ to find a path for each agent $a_i \in \mathcal{A}_k$ with the smallest arrival time $t_i^{(g)}$, which are similar in essence to prioritized offline MAPF algorithms (for example, Cooperative A*(Silver 2005)). Optimally-rational algorithms in PLAN-NEW (respectively, PLAN-ALL) with respect to flowtime can call any optimal offline MAPF algorithm at release time $r_{\mathcal{A}_k}$ to find a plan for all agents in $\mathcal{A}_k$ (respectively, $\mathcal{A}_{\leq k}$) with the smallest flowtime. Optimally-rational algorithms in PLAN-NEW (respectively, PLAN-ALL) with respect to makespan can call any optimal offline MAPF algorithm at release time $r_{\mathcal{A}_k}$ to find a plan for all agents in $\mathcal{A}_k$ (respectively, $\mathcal{A}_{\leq k}$) with the smallest makespan. Note that, unlike the above examples, optimally-rational algorithms with respect to makespan can alternatively find a plan for agents in $\mathcal{A}^C$ that does not necessarily have the smallest makespan but still results in a plan for all planned agents with the smallest makespan (and is thus optimally-rational).

4 Feasibility and SEQUENCE

We now show that, unlike (offline) MAPF (in which agents are not removed upon arrival at goal vertices), all online MAPF instances are solvable. We prove the statement by describing the following naive online MAPF algorithm **SEQUENCE** that routes agents one after another in sequence.

SEQUENCE plans a path for one agent at a time in increasing order of the indices of the agents (thus also non-decreasing order of the release times of the agents). Specifically, it first plans a path π_1 for agent a_1 at time step r_1, when the agent is revealed, such that the agent starts from vertex s_1 at time step $t_1^{(s)} = r_1$, moves along the shortest path from vertex s_1 to vertex g_1 in graph G, and arrives at vertex g_1 at time step $t_1^{(g)} = t_1^{(s)} + dist_1$. Then, for each $i = 2, \ldots, m$, it plans a path π_i for agent a_i at time step r_i such that the agent starts at time step $t_i^{(s)} = \max(r_i, t_{i-1}^{(g)})$, moves along the shortest path from vertex s_i to vertex g_i in graph G, and arrives at vertex g_i at time step $t_i^{(g)} = t_i^{(s)} + dist_i$. In other words, each such agent starts routing only when the previous agent has finished routing and been removed.

Observation 5. *All online MAPF instances are solvable, and SEQUENCE solves them.*

Reason. The shortest path computation of paths π_i for all $i \in [m]$ succeeds because graph G is connected. Each agent thus arrives at its goal vertex at a finite time step. Also, the resulting paths are collision-free since no two agents are in graph G at the same time step. □

4.1 Competitive Ratio Upper Bounds

We follow the standard definition of competitive ratio (Borodin and El-Yaniv 2005). Let the cost of online algorithm *ALG* for an input sequence σ (in our case, the sequence of agents) be $C_{ALG}(\sigma)$ with respective to a given objective function and the cost of an optimal offline algorithm *OPT* that knows the entire input sequence σ a priori (in our case, has full controllability at time step 0 of all agents $a_1, \ldots, a_m$ that might be revealed in the future) be $C_{OPT}(\sigma)$.

Definition 6 (Competitive Ratio)**.** An online algorithm *ALG* is α-*competitive* or has a *competitive ratio* of α if, for all input sequence σ and some constant δ, $C_{ALG}(\sigma) \leq \alpha C_{OPT}(\sigma) + \delta$.

We now derive upper bounds on the competitive ratio for SEQUENCE with respect to flowtime and makespan in the following theorems, respectively.

Theorem 7. *SEQUENCE achieves a competitive ratio of $\mathcal{O}(m)$ with respect to flowtime.*

Proof. We first show by induction on i that the service time $t_i^{(g)} - r_i$ of each agent a_i is no larger than $\sum_{j\in[i]} dist_j$. The statement holds trivially for agent a_1. Assume that its holds for agent a_{i-1}. The service time of agent a_i is thus

$$\begin{aligned} t_i^{(g)} - r_i &= t_i^{(s)} + dist_i - r_i \\ &= \max(r_i, t_{i-1}^{(g)}) + dist_i - r_i \\ &= \max(0, t_{i-1}^{(g)} - r_i) + dist_i \\ &\overset{\text{definition}}{\leq} \max(0, t_{i-1}^{(g)} - r_{i-1}) + dist_i \\ &= t_{i-1}^{(g)} - r_{i-1} + dist_i \\ &\overset{\text{induction}}{\leq} \sum_{j\in[i-1]} dist_j + dist_i = \sum_{j\in[i]} dist_j. \end{aligned} \quad (1)$$

The statement thus holds also for agent a_i. The flowtime of the plan for all agents is thus no larger than $\sum_{i\in[m]} \sum_{j\in[i]} dist_j \leq m \sum_{j\in[m]} dist_j$. Since the optimal flowtime is no smaller than $\sum_{j\in[m]} dist_j$, the theorem follows. □

Theorem 8. *SEQUENCE achieves a competitive ratio of $\mathcal{O}(m)$ with respect to makespan.*

Proof. Let r_{n_K} be the latest release time with $r_{n_K} > r_1 + \sum_{j\in[n_K-1]} dist_j$ or $r_{n_K} = r_1$ if there is no such release time. According to Equation (1), SEQUENCE computes a solution with makespan no larger than $r_{n_K} + \sum_{j\in[n_K,m]} dist_j \leq r_{n_K} + m \max_{j\in[n_K,m]} dist_j$. Since the optimal makespan is no smaller than $\max_{j\in[n_K,m]}(r_{n_K} + dist_j)$, the theorem follows. □

We show in Section 6.4 that the competitive ratio for SEQUENCE is infinite with respect to latency.

5 Rationality and Competitive Ratio Upper Bounds

The $\mathcal{O}(m)$ upper bounds on the competitive ratio for the naive algorithm SEQUENCE shown in Theorems 7 and 8 set a baseline for all online MAPF algorithms about what behavior (quality of the plans returned by the algorithms) is rational. This analysis also inspires the characterization of *rational* algorithms that are guaranteed to result in a solution quality (asymptotically) no worse than SEQUENCE, which extends the notion of optimally-rational algorithms. Note that our definition of rationality is significantly different from that in the optimization and economics literature.

This version of the paper is intended to update the version published in the Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICASP 2021). The definition of *makespan* (Section 2) has been corrected from $\max_{a_i\in\mathcal{A}}(t_i^{(g)} - r_i)$ to $\max_{a_i\in\mathcal{A}} t_i^{(g)}$. Other parts of the paper are not affected.

Definition 9 (Rationality)**.** An online MAPF algorithm is *rational* if and only if, at each release time $r_{\mathcal{A}_k}$, $\forall k \in [K]$, the plan for all revealed agents (agents in $\mathcal{A}_{\leq k}$) has flowtime no larger than $|\mathcal{A}_{\leq k}| \sum_{a_i \in \mathcal{A}_{\leq k}} dist_i$ and makespan no larger than $r_{n_k} + \sum_{i \in [n_k, m_k]} dist_i$ where $m_k = \arg\max_i a_i \in \mathcal{A}_{\leq k}$ and $r_{n_k} \leq r_{\mathcal{A}_k}$ is the latest release time of agent a_{n_k} with $r_{n_k} > r_1 + \sum_{i \in [n_k - 1]} dist_i$ or $r_{n_k} = r_1$ if there is no such release time.

The main idea behind Definition 9 is to set upper bounds that quantify how badly an online MAPF algorithm could perform at each release time $r_{\mathcal{A}_k}$ with respect to both flowtime and makespan so that it still results in a solution quality that is asymptotically no worse than SEQUENCE. Trivially, SEQUENCE is rational by using a similar argument as in the proofs of Theorems 7 and 8 for each release time $r_{\mathcal{A}_k}$. The flowtime term in Definition 9 can be rewritten with respect to latency. However, we show in Section 6.4 that the competitive ratio for all rational algorithms is (at least) infinite with respect to latency and thus do not study its upper bound in this section.

5.1 Rationalization

Not all online MAPF algorithms are rational. However, one can *rationalize* any online MAPF algorithm in PLAN-NEW and PLAN-ALL by adding a simple subroutine to it as follows: At each release time $r_{\mathcal{A}_k}, \forall k \in [K]$, the online MAPF algorithm calls the subroutine at the end of its computation that checks whether the resulting plan for all agents in $\mathcal{A}_k$ respects the upper bounds set by Definition 9. If so, it changes the plan so that the agents move to their goal vertices one after another in increasing order of their indices, as in SEQUENCE, starting at time step $t^{(s)}_{\mathcal{A}_k}$ that is the maximum of $r_{\mathcal{A}_k}$ and the makespan of the (old) computed plan at the previous release time $r_{\mathcal{A}_{k-1}}$ for $k > 1$ and time step $t^{(s)}_{\mathcal{A}_k} = r_1$ for $k = 1$. The key idea is that, after each computation, it checks whether the resulting plan (for the set of agents that have paths) respects the upper bounds set by Definition 9 and, if not, switches to the same behavior as that of SEQUENCE to guarantee that the plan is asymptotically no worse than the plan (for the same set of agents) in SEQUENCE. We have omitted the discussion of rationalization for PLAN-NEW-SINGLE above, which can be achieved by adding a similar subroutine after each single-agent path-finding computation that compares the flowtime and makespan of the resulting plan to those in SEQUENCE.

5.2 Rationality of Optimally-Rational Algorithms

Intuitively, online MAPF algorithms that look reasonable, for example, ones that are optimally-rational, are rational (without rationalization). We prove the following theorem, even though the name "optimal rationality" has already suggested it.

Theorem 10. *Optimally-rational online MAPF algorithms under any given controllability assumption with respect to either flowtime or makespan are rational.*

Proof. We consider an arbitrary optimally-rational online MAPF algorithm in PLAN-NEW-SINGLE with respect to either flowtime or makespan. We show that, at each release time $r_{\mathcal{A}_k}$, $\forall k \in [K]$, the plan for all revealed agents has flowtime no larger than $|\mathcal{A}_{\leq k}| \sum_{a_i \in \mathcal{A}_{\leq k}} dist_i$ and makespan no larger than $r_{n_k} + \sum_{i \in [n_k, m_k]} dist_i$ and the algorithm is thus rational (Definition 9). We first show by induction on i that the service time $t^{(g)}_i - r_i$ of each agent a_i is no larger than $\sum_{j \in [i]} dist_j$. The statement holds trivially for agent a_1 for such an algorithm. Assume that it holds for agents $a_2, \ldots, a_{i-1}$. The service time of agent a_i is thus

$$\begin{aligned}
t^{(g)}_i - r_i = \quad & t^{(s)}_i + dist_i - r_i \\
\overset{\text{opt. rat.}}{\leq} \quad & \max(r_i, \max_{j \in [i-1]} t^{(g)}_j) + dist_i - r_i \\
\leq \quad & \max(0, \max_{j \in [i-1]} (t^{(g)}_j - r_i)) + dist_i \\
\overset{\text{definition}}{\leq} \quad & \max(0, \max_{j \in [i-1]} (t^{(g)}_j - r_j)) + dist_i \\
\overset{\text{induction}}{\leq} \quad & \max(0, \max_{j \in [i-1]} \sum_{j' \in [j]} dist_{j'}) + dist_i \\
= \quad & \sum_{j \in [i-1]} dist_j + dist_i = \sum_{j \in [i]} dist_j
\end{aligned}$$

for any optimally-rational online MAPF algorithm in PLAN-NEW-SINGLE with respect to either flowtime or makespan. The statement thus holds also for agent a_i. The plan for all revealed agents at any release time $r_{\mathcal{A}_k}$ thus has flowtime F_k no larger than $\sum_{i \in \mathcal{A}_{\leq k}} \sum_{j \in [i]} dist_j \leq |\mathcal{A}_{\leq k}| \sum_{j \in \mathcal{A}_{\leq k}} dist_j$ and makespan M_k no larger than $r_{n_k} + \sum_{j \in [n_k, m_k]} dist_j$. For any optimally-rational online MAPF algorithm in PLAN-NEW or PLAN-ALL with respect to either flowtime or makespan, the plan for all revealed agents at each release time $r_{\mathcal{A}_k}$ has flowtime no larger than F_k and makespan no larger than M_k (or it is not optimally-rational otherwise). Rationality is thus maintained at each release time $r_{\mathcal{A}_k}$. □

5.3 Competitive Ratio Upper Bounds

We show that all rational online MAPF algorithms perform asymptotically no worse than SEQUENCE with respect to both flowtime and makespan.

Theorem 11. *All rational online MAPF algorithms achieve a competitive ratio of $\mathcal{O}(m)$ with respect to flowtime.*

Proof. According to Definition 9, the plan computed by any rational online MAPF algorithm at the latest release time $r_{\mathcal{A}_K} = r_m$ has flowtime no larger than $|\mathcal{A}_{\leq K}| \sum_{a_i \in \mathcal{A}_{\leq K}} dist_i = m \sum_{i \in [m]} dist_i$. Since the optimal flowtime is no smaller than $\sum_{i \in [m]} dist_i$, the theorem follows. □

Theorem 12. *All rational online MAPF algorithms achieve a competitive ratio of $\mathcal{O}(m)$ with respect to makespan.*

Proof. According to Definition 9, the plan computed by any rational online MAPF algorithm at the latest release time $r_{\mathcal{A}_K} = r_m$ has makespan no larger than $r_{n_K} + \sum_{i \in [n_K, m]} dist_i \leq r_{n_K} + m \max_{i \in [n_K, m]} dist_i$. Since the optimal makespan is no smaller than $\max_{i \in [n_K, m]}(r_{n_K} + dist_i)$, the theorem follows. □

This version of the paper is intended to update the version published in the Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICASP 2021). The definition of *makespan* (Section 2) has been corrected from $\max_{a_i \in \mathcal{A}}(t^{(g)}_i - r_i)$ to $\max_{a_i \in \mathcal{A}} t^{(g)}_i$. Other parts of the paper are not affected.

6 Competitive Ratio Lower Bounds

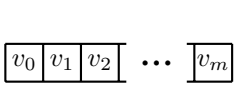
We show that all rational online MAPF algorithms in PLAN-NEW achieve a competitive ratio of at least $\Omega(m)$ with respect to both flowtime and makespan by constructing an online MAPF instance on a 4-neighbor 2D grid. Therefore, the $\mathcal{O}(m)$ upper bounds on the competitive ratio of rational online MAPF algorithms in PLAN-NEW that we derived in Theorems 11 and 12 are (asymptotically) tight even for online MAPF instances on 4-neighbor 2D grids. Consequently, we show that rational algorithms do not necessarily outperform irrational ones. We also derive lower bounds on the competitive ratio of rational online MAPF algorithms in PLAN-ALL with respect to flowtime and makespan. Finally, we show that all rational online MAPF algorithms have infinite competitive ratio with respect to latency.

6.1 Rational Algorithms in PLAN-NEW

The following theorems show that the competitive ratio of all rational online MAPF algorithms in PLAN-NEW, including optimally-rational ones, is at least $\Omega(m)$ even on 4-neighbor 2D grids (and even lines) with respect to both flowtime and makespan.

Theorem 13. *There exists an online MAPF instance on a 4-neighbor 2D grid for which any rational online MAPF algorithm in PLAN-NEW achieves a competitive ratio of $\Omega(m)$ with respect to flowtime.*

Proof. Consider an online MAPF instance on the 4-neighbor 2D grid shown in Figure 3(a) where an even number m of agents are given and agent a_i has release time $r_i = i - 1$, for all $i \in [m]$. Agent a_i has start vertex $s_i = v_0$ and goal vertex $g_i = v_m$ if i is odd and start vertex $s_i = v_m$ and goal vertex $g_i = v_0$ if i is even. Consider an arbitrary rational online MAPF algorithm in PLAN-NEW. At time step 0, the algorithm computes the only possible path for agent a_1 where it starts at time step 0, moves without waiting, and arrives at $g_1 = v_m$ at time step m since other paths violate the makespan upper bound in Definition 9. At time step 1, the algorithm computes the only possible path for agent a_2 where it starts at time step m when agent a_1 has arrived at g_1 (or it collides with agent a_1 otherwise), moves without waiting, and arrives at $g_2 = v_0$ at time step $2m$ since other paths violate the makespan upper bound in Definition 9 ($r_{n_2} = r_1$). We apply the argument to each agent a_i, $i \in [3, m]$, that it has to start after agent a_{i-1} has arrived at g_{i-1} (since the paths for all previously-revealed agents do not change for algorithms in PLAN-NEW) and moves to its goal vertex without waiting at any vertex to maintain rationality (Definition 9 with $r_{n_k} = r_1$ for release time $r_{\mathcal{A}_k}, \forall k \in [K], K = m$). The resulting flowtime is $m+(2m-1)+(3m-2)+\ldots+(m^2-m+1) = \frac{1}{2}m^3+\frac{1}{2}m$. The optimal plan lets agents a_i start at their release times r_i if i is odd and at time steps $2m-3+\frac{i}{2}$ if i is even (from time step $2m-2$ on when the last agent a_{m-1} with an odd index has arrived at its goal vertex). All agents move to their goal vertices without waiting once they start, resulting in service time m for all odd numbers i and $3m-2-\frac{i}{2}$ for all even numbers i. The optimal flowtime is thus $\frac{15}{8}m^2 - \frac{5}{4}m$. The theorem thus follows. □



(a) Theorem 13.

(b) Theorem 18.

Figure 3: Online MAPF instances used for theorems.

Theorem 14. *There exists an online MAPF instance on a 4-neighbor 2D grid for which any rational online MAPF algorithm in PLAN-NEW achieves a competitive ratio of $\Omega(m)$ with respect to makespan.*

Proof. Using the same online MAPF instance and argument as in the proof of Theorem 13, any rational online MAPF algorithm in PLAN-NEW results in makespan m^2. The optimal plan (with respect to both flowtime and makespan) has makespan $\frac{7}{2}m - 3$. The theorem thus follows. □

Theorems 13 and 14 thus show that the upper bounds that we derived in Theorems 11 and 12 for flowtime and makespan, respectively, are asymptotically tight for all rational online MAPF algorithms in PLAN-NEW, yielding the following corollaries.

Corollary 15. *All rational online MAPF algorithms in PLAN-NEW are $\Theta(m)$-competitive with respect to flowtime even on 4-neighbor 2D grids.*

Corollary 16. *All rational online MAPF algorithms in PLAN-NEW are $\Theta(m)$-competitive with respect to makespan even on 4-neighbor 2D grids.*

6.2 Irrational Algorithms

A counter-intuitive insight we obtain from the above theorems is that rational online MAPF algorithms do not necessarily outperform the irrational ones and rationalization could harm the solution quality.

Observation 17. *There exists an online MAPF instance on a 4-neighbor 2D grid for which an irrational online MAPF algorithm in PLAN-NEW-SINGLE outperforms any rational and thus rationalized online MAPF algorithms in PLAN-NEW with respect to both flowtime and makespan.*

Reason. For the online MAPF instance used in the proofs of Theorems 13 and 14, a dummy online MAPF algorithm that imitates the optimal offline algorithm, which can be viewed as in PLAN-NEW-SINGLE, is irrational and outperforms any rational online MAPF algorithm in PLAN-NEW. □

However, it is "irrational" and impossible for one to design such a perfect irrational online MAPF algorithm, for example, one that delays agent a_2 by m time steps in the above example, that imitates the behavior of the optimal offline algorithm in practice without any knowledge of future arrival of agents. Also note that the above reasoning does not carry over to rational online MAPF algorithms in PLAN-ALL.

6.3 Rational Algorithms in PLAN-ALL

We now construct an online MAPF instance on a 4-neighbor 2D grid for which all rational algorithms in PLAN-ALL achieve a constant competitive ratio.

This version of the paper is intended to update the version published in the Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICASP 2021). The definition of *makespan* (Section 2) has been corrected from $\max_{a_i \in \mathcal{A}}(t_i^{(g)} - r_i)$ to $\max_{a_i \in \mathcal{A}} t_i^{(g)}$. Other parts of the paper are not affected.

Theorem 18. *There exists an online MAPF instance on a 4-neighbor 2D grid for which any rational online MAPF algorithm achieves a competitive ratio of at least 4/3 with respect to flowtime.*

Proof. Consider an online MAPF instance on the 4-neighbor 2D grid shown in Figure 3(b) where two agents a_1 and a_2 are given. Agent a_1 has start vertex v_1, goal vertex v_4, and release time 0. Consider an arbitrary rational online MAPF algorithm. At time step 0, the algorithm computes one of the two possible paths for agent a_1 where the agent starts at time step 0, moves without waiting, and arrives at $g_1 = v_4$ at time step 2 according to Definition 9. If agent a_1 chooses to go through vertex v_2, we construct agent a_2 with start vertex v_2, goal vertex v_1, and release time 1. Otherwise (agent a_1 chooses to go through vertex v_3), we construct agent a_2 with start vertex v_3, goal vertex v_1, and release time 1. In either case, agent a_2 starts at time step 2. The resulting flowtime is 4. The optimal plan has flowtime 3. The theorem thus follows. □

Theorem 19. *There exists an online MAPF instance on a 4-neighbor 2D grid for which any rational online MAPF algorithm achieves a competitive ratio of at least 3/2 with respect to makespan .*

Proof. Using the same online MAPF instance and argument as in the proof of Theorem 18, any rational online MAPF algorithm results in makespan 3. The optimal plan has makespan 2. The theorem thus follows. □

6.4 Infinite Competitive Ratio for Latency

We have not performed any analysis for latency so far since we have the following observation from the above example.

Observation 20. *All rational online MAPF algorithms have infinite competitive ratio with respect to latency even on 4-neighbor 2D grids.*

Reason. Using the same online MAPF instance and argument as in the proof of Theorem 18, any rational online MAPF algorithm results in latency 1. The optimal plan has latency 0. Therefore, the competitive ratio of any rational online MAPF algorithm is (at least) infinite with respect to latency even on 4-neighbor 2D grids. □

7 Conclusions

We conducted a theoretical study of online MAPF for the first time. Our results suggest that, if rerouting is disallowed, then planning for multiple agents is asymptotically (only) as effective as planning for one agent at a time and acting optimally rationally is asymptotically (only) as effective as acting rationally, which is also asymptotically (only) as effective as following the naive algorithm SEQUENCE. However, allowing rerouting can potentially result in high effectiveness, as indicated by the gap between the competitive ratio upper and lower bounds.

Future work includes (1) developing a rational online MAPF algorithm in PLAN-ALL that achieves the current competitive ratio lower bound (thus proving that the bound is tight for it) or tightening the bounds further and (2) analyzing the online MAPF variant where probabilistic models of future arrivals of agents are given.

Appendix: Proof of Theorem 1

Proof. Similar to the proof of Theorem 3 in Ma et al. (2016), we construct an online MAPF instance that has a solution with makespan three if and only if a given ≤3,=3-SAT instance is satisfiable. For each variable X_i in the ≤3,=3-SAT instance, we construct two "literal" agents, a_{iT} and a_{iF}, with start vertices s_{iT} and s_{iF} and goal vertices t_{iT} and t_{iF}, respectively. All literal agents have release time zero. For each literal agent, we construct two paths to get to its goal vertex in three time steps: a "shared" path, namely $\langle s_{iT}, u_{iT}, v_i, t_{iT}\rangle$ for a_{iT} and $\langle s_{iF}, u_{iF}, v_i, t_{iF}\rangle$ for a_{iF}, and a "private" path, namely $\langle s_{iT}, w_{iT}, x_{iT}, t_{iT}\rangle$ for a_{iT} and $\langle s_{iF}, w_{iF}, x_{iF}, t_{iF}\rangle$ for a_{iF}. The shared paths for a_{iT} and a_{iF} intersect at vertex v_i. Only one of the two paths can thus be used if a makespan of three is to be achieved. Sending literal agent a_{iT} (or a_{iF}) along the shared path corresponds to assigning *True* (or *False*) to X_i in the ≤3,=3-SAT instance. For each clause C_j in the ≤3,=3-SAT instance, we construct a "clause" agent a_j with start vertex c_j, goal vertex d_j, and release time zero. It has multiple (but at most three) "clause" paths to get to its goal vertex in three time steps, which have a one-to-one correspondence to the literals in C_j. Every literal X_i (or $\overline{X_i}$) can appear in at most two clauses. If C_j is the first clause that it appears in, then the clause path is $\langle c_j, w_{iT}, b_j, d_j\rangle$ (or $\langle c_j, w_{iF}, b_j, d_j\rangle$). If C_j is the second clause that it appears in, a vertex α_j is introduced and the clause path is instead $\langle c_j, \alpha_j, x_{iT}, d_j\rangle$ (or $\langle c_j, \alpha_j, x_{iF}, d_j\rangle$). The clause path of each C_j with respect to any literal in that clause and the private path of the literal intersect. Only one of the two paths can thus be used if a makespan of three is to be achieved. A visualized example of the reduction can be found in Ma et al. (2016). Suppose that a satisfying assignment to the ≤3,=3-SAT instance exists. Then, a solution with makespan three is obtained by sending literal agents of true literals along their shared paths, the other literal agents along their private paths, and clause agents along the clause paths corresponding to one of the true literals in those clauses. Conversely, suppose that a solution with makespan three exists. Then, each clause agent traverses the clause path corresponding to one of the literals in that clause, and the corresponding literal agent traverses its shared path. Since the agents of a literal and its complement cannot both use their shared path if a makespan of three is to be achieved, we can assign *True* to every literal whose agent uses its shared path without assigning *True* to both the uncomplemented and complemented literals. If the agents of both literals use their private paths, we can assign *True* to any one of the literals and *False* to the other one. A solution to the online MAPF instance with makespan three thus yields a satisfying assignment to the ≤3,=3-SAT instance.

To summarize, the online MAPF instance has a solution with makespan three if and only if the ≤3,=3-SAT instance is satisfiable. Also, the online MAPF instance cannot have a solution with makespan smaller than three and always has a solution with makespan four, even if the ≤3,=3-SAT instance is unsatisfiable. For any $\epsilon > 0$, any approximation algorithm for online MAPF with ratio $4/3 - \epsilon$ thus computes a solution with makespan three whenever the ≤3,=3-SAT instance is satisfiable and therefore solves ≤3,=3-SAT. □

This version of the paper is intended to update the version published in the Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICASP 2021). The definition of *makespan* (Section 2) has been corrected from $\max_{a_i \in \mathcal{A}}(t_i^{(g)} - r_i)$ to $\max_{a_i \in \mathcal{A}} t_i^{(g)}$. Other parts of the paper are not affected.

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.

Borodin, A.; and El-Yaniv, R. 2005. *Online Computation and Competitive Analysis*. Cambridge University Press.

Boyarski, E.; Felner, A.; Stern, R.; Sharon, G.; Tolpin, D.; Betzalel, O.; and Shimony, S. E. 2015. ICBS: Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding. In *IJCAI*, 740–746.

Cohen, L.; Greco, M.; Ma, H.; Hernandez, C.; Felner, A.; Kumar, T. K. S.; and Koenig, S. 2018. Anytime Focal Search with Applications. In *IJCAI*, 1434–1441.

Das, A.; Gollapudi, S.; Kim, A.; Panigrahi, D.; and Swamy, C. 2018. Minimizing latency in online ride and delivery services. In *WWW*, 379–388.

Dresner, K.; and Stone, P. 2008. A Multiagent Approach to Autonomous Intersection Management. *Journal of Artificial Intelligence Research* 31: 591–656.

Erdem, E.; Kisa, D. G.; Oztok, U.; and Schueller, P. 2013. A General Formal Framework for Pathfinding Problems with Multiple Agents. In *AAAI*, 290–296.

Gange, G.; Harabor, D.; and Stuckey, P. J. 2019. Lazy CBS: Implicit Conflict-Based Search Using Lazy Clause Generation. In *ICAPS*, 155–162.

Ho, F.; Salta, A.; Geraldes, R.; Goncalves, A.; Cavazza, M.; and Prendinger, H. 2019. Multi-agent Path Finding for UAV Traffic Management. In *AAMAS*, 131–139.

Kou, N. M.; Peng, C.; Ma, H.; Kumar, T. S.; and Koenig, S. 2020. Idle time optimization for target assignment and path finding in sortation centers. In *AAAI*, volume 34, 9925–9932.

Lam, E.; Le Bodic, P.; Harabor, D.; and Stuckey, P. J. 2019. Branch-and-Cut-and-Price for Multi-Agent Pathfinding. In *IJCAI*, 1289–1296.

Li, J.; Chen, Z.; Zheng, Y.; Chan, S.-H.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2021. Scalable Rail Planning and Replanning: Winning the 2020 Flatland Challenge. In *ICAPS*.

Li, J.; Felner, A.; Boyarski, E.; Ma, H.; and Koenig, S. 2019a. Improved Heuristics for Multi-Agent Path Finding with Conflict-Based Search. In *IJCAI*, 442–449.

Li, J.; Gange, G.; Harabor, D.; Stuckey, P. J.; Ma, H.; and Koenig, S. 2020. New Techniques for Pairwise Symmetry Breaking in Multi-Agent Path Finding. In *ICAPS*, 193–201.

Li, J.; Harabor, D.; Stuckey, P. J.; Felner, A.; Ma, H.; and Koenig, S. 2019b. Disjoint Splitting for Conflict-Based Search for Multi-Agent Path Finding. In *ICAPS*, 279–283.

Luna, R.; and Bekris, K. E. 2011. Push and Swap: Fast Cooperative Path-finding with Completeness Guarantees. In *IJCAI*, 294–300.

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

Ma, H.; Hönig, W.; Kumar, T. K. S.; Ayanian, N.; and Koenig, S. 2019b. Lifelong Path Planning with Kinematic Constraints for Multi-Agent Pickup and Delivery. In *AAAI*, 7651–7658.

Ma, H.; and Koenig, S. 2017. AI Buzzwords Explained: Multi-Agent Path Finding (MAPF). *AI Matters* 3(3): 15–19.

Ma, H.; Li, J.; Kumar, T. K. S.; and Koenig, S. 2017a. Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks. In *AAMAS*, 837–845.

Ma, H.; Tovey, C.; 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 *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: Preliminary Results. In *AAMAS*, 2004–2006.

Ma, H.; Yang, J.; Cohen, L.; Kumar, T. K. S.; and Koenig, S. 2017b. Feasibility Study: Moving Non-Homogeneous Teams in Congested Video Game Environments. In *AIIDE*, 270–272.

Mohanty, S.; Nygren, E.; Eichenberger, C.; Baumberger, C.; Egli, A.; Spigler, G.; Watson, J.; Laurent, F.; Scheller, C.; Bhattacharya, N.; Sartoretti, G.; Sturm, I.; and Koenig, S. 2020. FLATLAND NeurIPS 2020 Competition: Multi-Agent Reinforcement Learning on Trains. Online.

Phillips, M.; and Likhachev, M. 2011. SIPP: Safe Interval Path Planning for Dynamic Environments. In *ICRA*, 5628–5635.

Salzman, O.; and Stern, R. 2020. Research Challenges and Opportunities in Multi-Agent Path Finding and Multi-Agent Pickup and Delivery Problems. In *AAMAS*, 1711–1715.

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

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

Silver, D. 2005. Cooperative Pathfinding. In *AIIDE*, 117–122.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T. K. S.; Boyarski, E.; and Bartak, R. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *SoCS*, 151–159.

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

Surynek, P.; Felner, A.; Stern, R.; and Boyarski, E. 2016. Efficient SAT Approach to Multi-Agent Path Finding Under the Sum of Costs Objective. In *ECAI*, 810–818.

Švancara, J.; Vlk, M.; Stern, R.; Atzmon, D.; and Barták, R. 2019. Online Multi-Agent Pathfinding. In *AAAI*, 7732–7739.

Tovey, C. A. 1984. A Simplified NP-Complete Satisfiability Problem. *Discrete Applied Mathematics* 8: 85–90.

Wang, K.; and Botea, A. 2011. MAPP: A Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees. *Journal of Artificial Intelligence Research* 42: 55–90.

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

Yu, J. 2015. Intractability of optimal multirobot path planning on planar graphs. *IEEE Robotics and Automation Letters* 1(1): 33–40.

Yu, J.; and LaValle, S. M. 2013a. Planning Optimal Paths for Multiple Robots on Graphs. In *ICRA*, 3612–3617.

Yu, J.; and LaValle, S. M. 2013b. Structure and Intractability of Optimal Multi-robot Path Planning on Graphs. In *AAAI*, 1444–1449.

This version of the paper is intended to update the version published in the Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICASP 2021). The definition of *makespan* (Section 2) has been corrected from $\max_{a_i \in \mathcal{A}}(t_i^{(g)} - r_i)$ to $\max_{a_i \in \mathcal{A}} t_i^{(g)}$. Other parts of the paper are not affected.