To appear in Int'l Conference on Autonomous Agents and Multi-agent Systems (AAMAS), May 2020.

Explainable Multi Agent Path Finding

Shaull Almagor*
Computer Science Department, Technion
Haifa, Israel
shaull@cs.technion.ac.il

Morteza Lahijanian†
University of Colorado Boulder
Boulder, Colorado
morteza.lahijanian@colorado.edu

ABSTRACT

Multi Agent Path Finding (MAPF) is the problem of planning paths for agents to reach their targets from their start locations, such that the agents do not collide while executing the plan. In safety-critical systems, the plan is typically checked by a human supervisor, who decides on whether to allow its execution. In such cases, we wish to convince the human that the plan is indeed collision free.

To this end, we propose an *explanation scheme* for MAPF, which bases explanations on simplicity of visual verification by human's cognitive process. The scheme decomposes a plan into segments such that within each segment, the paths of the agents are disjoint. Then, we can convince the supervisor that the plan is collision free using a small number of images (dubbed an *explanation*). In addition, we can measure the simplicity of a plan by the number of segments required for the decomposition. We study the complexity of algorithmic problems that arise by the explanation scheme, as well as the tradeoff between the length (makespan) of a plan and its minimal decomposition. We also provide experimental results of our scheme both in a continuous and in a discrete setting.

KEYWORDS

multi-agent systems, path planning, explainability, MAPF, path finding, motion planning

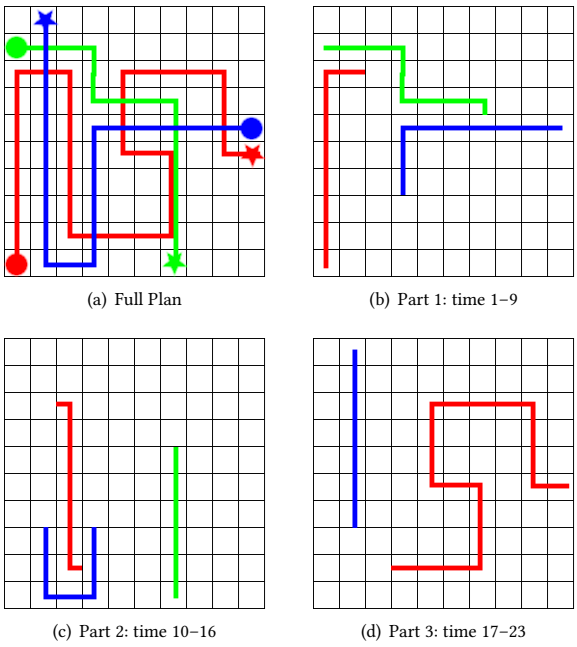
ACM Reference Format:
Shaull Almagor and Morteza Lahijanian. 2020. Explainable Multi Agent Path Finding . In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

Multi Agent Path Finding (MAPF)

MAPF is a fundamental problem in AI, in which the goal is to plan paths for several agents to reach their targets, such that paths can be taken simultaneously, without the agents colliding. Applications of MAPF are ubiquitous in any area where several moving agents are involved, such as air-traffic control, UAVs, warehouse robots, autonomous cars, robotics, etc.

Unfortunately, most variants of MAPF are intractable. However, the importance of this problem has generated a significant body of work over the past decade [4, 7, 13, 24, 29, 30, 33], dealing with various aspects of the problem and suggesting increasingly scalable solutions.



(a) Full Plan (b) Part 1: time 1–9 (c) Part 2: time 10–16 (d) Part 3: time 17–23

Figure 1: A plan for three agents in a 10×10 grid. The circles and stars mark the initial and goal locations for the agents, respectively. Figure (a) shows the full plan, and Figures (b), (c), and (d) show a disjoint decomposition.

Planning for a Human Supervisor

In many safety critical applications (e.g., air traffic control, hazardous-materials warehouses), planning is not fully automatic, and the plan is only suggested to a human supervisor, who may act upon it. In such settings, the plan has to be presented to the supervisor in some humanly-understandable manner. In particular, the presentation should enable the supervisor to understand the paths taken by the agents, and to easily verify that the agents do not collide, as otherwise the supervisor would not necessarily trust the plan. We call such a representation an *explanation* of the plan.

This former requirement is easy to meet – one can simply depict the paths taken by the agents. Unfortunately, the paths suggested by the plan may well intersect, even when the agents do not collide (see Figure 1a). Thus, a single picture of the paths does not suffice for the supervisor to understand the plan. At the other extremity,

*Shaull Almagor has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 837327.
†Morteza Lahijanian has received funding from the University of Colorado Boulder.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

one could display a sequence of images or a video of the agents taking their paths. While it is possible to use video in order to verify the plan, it is difficult for a human observer to verify that no two agents collide, unless the video is played very slowly, or the sequence of images is examined very carefully.

In this work, we suggest a novel approach for explaining plans to human supervisors by means of *disjoint decompositions*. Intuitively, we decompose the plan for the agents into time segments such that, within each time segment, the paths taken by the agents are disjoint. Then, we display each time segment as a separate picture. In Figure 1, we demonstrate such a decomposition of a plan into three parts.

Remark 1.1. *The reader may notice that verifying no two lines intersect in a picture can be done very quickly. Indeed, the identification of line intersections is done very early in the cognitive process (namely in the primary visual cortex [18, 35]).*

Remark 1.2. *Formally, our framework in Section 2 places no constraint on the environment. However, since we output pictures of the paths, our explanation scheme is more relevant to two dimensional plans. See Section 6 for a discussion about three dimensional explanations.*

Crucially, our explanations can be quickly verified, and so can be used by a supervisor in real-time to make a decision.

What is an Explanation?

The notion of "explanation" is not well defined and is hugely dependent on the problem at hand and on the user to whom the explanation is given. For example, in [23], visualization is used to explain the result of certain Machine Learning algorithms, that often come up with complicated classifiers. In this case, the explanation is given to a human, and therefore visualization is helpful. In contrast, in [2], o-minimal sets are given as an explanation for the non-termination of linear loops. These explanations are useful, since it is decidable to check their correctness, whereas the general non-termination problem is not known to be decidable. However, actually checking the correctness is not necessarily tractable, so the user in this case is not a human, but rather a computer with unlimited time. This work is closer to [23] in that the user to whom we explain is a human, but instead of classification, the problem at hand is MAPF.

In order to make the notion of explanation slightly more concrete, we view it in the following way. Consider a decision problem P. An *explanation scheme* for P is a mechanism that outputs, for a given input I, some information called an explanation, or outputs that no explanation is found. Then, we can reason about three properties of the explanation scheme:

(1) Soundness: If an explanation exists, then I is a yes-instance.
(2) Completeness: If I is a yes-instance, then an explanation exists.
(3) Simplicity: An explanation is easy to find (if it exists) and to verify.

Note that the simplicity requirement is not formal; it is rather context-dependent. If the scheme is both sound and complete, then the explanations are in fact *proofs*, and if in addition they are simple, then the problem P itself would be simple to solve a priori. Thus, these properties hold simultaneously only in very naive problems. Challenge arises when different combinations of the above properties are considered with their *quantitative* tuning for explanations.

For example, in this work, our explanation scheme is sound but has a quantitative tradeoff between simplicity and completeness. That is, on one hand, the fewer plan segments (pictures) we output, the simpler the explanation is, but on the other hand, we can explain fewer plans using just a few pictures, so the scheme is less complete. If we allow an arbitrary number of pictures, we can explain any plan (so the scheme is complete), but the explanations are no longer simple.

Related Work

In recent years, there has been significant effort towards providing explanations for problems in AI and in machine learning. Such works are sometimes grouped under the *Explainable Artificial Intelligence (XAI)* title [10, 17]. The goal is to provide some meaningful interpretation for the results of algorithms, such that users of the algorithms (although not necessarily human users, as discussed above) are able to gain insight into the result.

Concrete examples of explanations take on various forms. In [11], explanations are given by analyzing possible alternative plans, where the space of alternative plans is determined by certain "interesting properties" given by the user. Then, explanations of plans are contrasive, in that when a user asks e.g., "why does plan π start with A", then answer is given in terms of properties that are violated if A is not taken. In [19], explanations for plans are given by a minimal set of differences between the proposed plan and a plan suggested by the user. This type of explanation is often more detailed, as it can be given in terms of comparison with a given alternative. A more unified approach is taken in [15], where several types of explanation queries are allowed. There, the user may change the plan in certain ways, and the planner should explain why the original plan was better, or re-plan.

Paper Organization

In Section 2, we introduce our explanation scheme for MAPF, namely *vertex-disjoint decompositions*, and state the relevant algorithmic problems. In Section 3, we show that finding optimal explanations for existing plans can be done efficiently, whereas planning for MAPF problems with simple explanations is **NP-Complete**. In addition, we study the tradeoff between time-optimal plans and plans with simple explanations.

In Section 4, we consider the continuous version of explainable MAPF and suggest an approach to tackle it via discretization. We discuss the tradeoff that arises in this setting between explainability and degree of refinement. In Section 5, we present some experimental results, both in the discrete and the continuous settings. In particular, in Section 5.1, we demonstrate the practical difficulties that arise in implementing a search-based algorithm for planning with explanations. We conclude with future research directions in Section 6.

2 PROBLEM FORMULATION

We start by giving the basic definitions used throughout the paper, followed by the formulation of the problems at hand.

Consider a directed graph $G = \langle V, E \rangle$. A *path* in G is a sequence $\pi = v_1 \dots v_m$ such that for all $1 \leq i < m$ either $(v_i, v_{i+1}) \in E$ or $v_i = v_{i+1}$ (intuitively, agents are allowed to wait in place). Let $\pi_1 = v_1 \dots v_{m_1}$ and $\pi_2 = u_1 \dots u_{m_2}$ be paths in G. We say that π_1 and π_2 are *non-colliding* if the following hold:

(1) $v_i \neq u_i$ for all $1 \leq i \leq \min\{m_1, m_2\}$ (i.e., no vertex collisions)[1],
(2) $(v_i, v_{i+1}) \neq (u_{i+1}, u_i)$ for all $1 \leq i < \min\{m_1, m_2\}$ (i.e., no vertex swapping).

We say that π_1 and π_2 are *vertex disjoint* if

$$\{v_0, \dots, v_{m_1}\} \cap \{u_0, \dots, u_{m_2}\} = \emptyset.$$

We lift these definitions to a set of paths by requiring that they hold pairwise.

Let $P = \{\pi_1, \dots, \pi_k\}$ be a set of non-colliding paths of length at most T. A *vertex-disjoint decomposition* of P is an ordered list ℓ of natural numbers $1 = t_0 < t_1 < \dots < t_r = T + 1$ such that for every $0 < i \leq r$, the paths $\{\pi_j[t_{i-1}, t_i - 1]\}_{j=1}^{k}$ are vertex-disjoint (where $\pi_j[a, b]$ refers to the infix of π_j between indices a and b, and we cut-off the path if the indices are out of bound). We refer to r as the *index* of ℓ. The minimal decomposition index of a vertex-disjoint decomposition of P is referred to as the *index* of P.

Remark 2.1. *Consider a set of non-colliding paths P of length T, where the length of P is the maximal length of the paths in P. A trivial vertex-disjoint decomposition of P is the list $1, 2, \dots, T + 1$. Thus, a vertex-disjoint decomposition always exists, and the index of P is at most T.*

Given k agents on graph G and two lists $s_1, \dots, s_k$ and $g_1, \dots, g_k$ of source and goal vertices, respectively, a *plan* is a set of non-colliding paths $P = \{\pi_1, \dots, \pi_k\}$ such that π_i leads agent i from s_i to g_i for all $1 \leq i \leq k$. The classical *Multi Agent Path Finding (MAPF)* is as follows.

Problem 1 (MAPF). *Given a graph $G = \langle V, E \rangle$ and lists $s_1, \dots, s_k$ and $g_1, \dots, g_k$, where $s_i, g_i \in V$ for all $1 \leq i \leq k$, find a plan for the agents.*

Note that this is a search problem. The decision version of the problem asks whether there exists such a plan, and the threshold version asks whether there exists a plan of length at most T. In addition, the optimization version asks to find the minimal-length plan.

The introduction of explanations gives rise to two problems. The first problem is to compute a minimal decomposition of a given plan, which is formally stated below.

Problem 2. *Given a graph $G = \langle V, E \rangle$ and a set of non-colliding paths (i.e., a plan) P, compute a vertex-disjoint decomposition of P with minimal index.*

Recall that by Remark 2.1, a vertex-disjoint decomposition always exists, so Problem 2 is well defined.

While a solution for Problem 2 may give us the "best" explanation for a given plan, there may exist other (possibly "worse") plans that offer better explanations. Thus, we consider the more involved problem of planning with explanations in mind. The formal statement of this problem is as follows.

Problem 3. *Given a graph $G = \langle V, E \rangle$, lists $s_1, \dots, s_k$ and $g_1, \dots, g_k$ of source and goal vertices, and a parameter $r \in \mathbb{N}$, find a plan P for the agents with the minimal vertex-disjoint decomposition index of at most r, or answer that no such plan exists.*

3 EXPLANATIONS OF DISCRETE MAPF

In this section, we study vertex-disjoint decompositions. We start by addressing Problem 2.

3.1 Computing Minimal Vertex-Disjoint Decompositions

Our first result is that computing a minimal-disjoint decomposition can be done efficiently.

Theorem 3.1. *Problem 2 can be solved in polynomial time.*

Proof. Let $P = \{\pi_1, \dots, \pi_k\}$ be a given plan, and let T denote its length. Our algorithm for computing a minimal vertex-disjoint decomposition proceeds in iterations, as follows. Set $t_0 = 1$. Then, at iteration i, the algorithm computes the maximal number $t_i \leq T + 1$ such that $\{\pi_j[t_{i-1}, t_i]\}_{j=1}^{k}$ is a set of vertex-disjoint paths. That is, we greedily keep following the paths until there is a collision, and then we start a new segment in the decomposition. Clearly, this algorithm can be implemented in polynomial time since, in every iteration, we only need to maintain a set of disjoint vertices, and we need to make only a single pass on P.

We turn to prove the correctness of this algorithm. Let $1 = t_0 < \dots < t_r = T + 1$ be the decomposition found by our algorithm. Consider some r' and a decomposition $1 = t'_1 < \dots < t'_{r'} = T + 1$ of P. We show that $r' \geq r$.

To this end, we claim that, for every $0 \leq i \leq r'$, it holds that $t'_i \leq t_i$. That is, the "endpoints" of the decomposition found by our algorithm are at least as high as those of the primed decomposition. We prove this by inductions over i. For $i = 0$ this holds by definition since $t_0 = t'_0 = 1$. Next, consider $1 \leq i + 1 \leq r'$; then, by the induction hypothesis, $t'_i \leq t_i$. Now, by the definition of our algorithm, t_{i+1} is the maximum such that $\{\pi_j[t_i, t_{i+1}]\}_{j=1}^{k}$ is vertex disjoint. In particular, it must hold that $t'_{i+1} \leq t_{i+1}$, since $\{\pi_j[t'_i, t'_{i+1}]\}_{j=1}^{k}$ is vertex disjoint as well, and $t'_i \leq t_i$.

The inductive argument implies that $r' \geq r$. Therefore, our decomposition is indeed minimal. □

The tractability of Problem 2 implies that explaining the results of MAPF can be readily incorporated into existing MAPF solvers.

3.2 Planning for Explainability

In this section, we study the more involved Problem 3, where we wish to find a plan that admits a small decomposition. In order to state complexity results, we address the decision version of Problem 3, namely:

Problem 4. *Given a graph $G = \langle V, E \rangle$, lists $s_1, \dots, s_k$ and $g_1, \dots, g_k$, and a parameter $r \in \mathbb{N}$, decide whether there exists a plan P for the agents with index at most r.*

[1] Since we only consider indices up to $\min\{m_1, m_2\}$, then, intuitively, once the goal is reached, the agent "disappears".

Observe that when $r = 0$, Problem 4 amounts to deciding whether there are vertex-disjoint paths for the agents, which is **NP-Hard** in several settings. Specifically, by [25, 26], we have the following:

Lemma 3.2. *Problem 4 is **NP-Hard**, even under each of the following restrictions:*

- *G is undirected and planar [25]*
- *G is directed and $k = 2$ (i.e., there are only two agents) [26]*

To provide a matching upper bound, we have the following:

Lemma 3.3. *Problem 4 is in **NP**.*

Proof. In [28], it is shown[2] that the shortest plan has length $B = O(n^3)$ (where $n = |V|$). Thus, if $r > B$, then deciding whether there is a plan with index at most r is equivalent to deciding whether there exists a plan of length at most r at all. Indeed, by splitting the plan into r segments of a single timestep, we obtain a decomposition. Since the latter problem is clearly in **NP**, this case is handled.

Next, we consider the case where $r < B$. A priori, it could be the case that, within each decomposition segment, the paths are very long (i.e., super-polynomial), rendering the plan very long. We show that this is not the case.

Consider a segment in a decomposition of a plan. Since the agents are allowed to stay in place, we can assume that in all paths within the segment, the only cycles are caused by staying in place. Indeed, replacing larger cycles by staying in place will keep the paths disjoint.

Next, observe that agents gain nothing by staying in place within a segment unless this is done so that other agents can finish traversing their (disjoint) path. Thus, staying in place for all agents can be delayed to the last vertex of the segment. Finally, we can omit any transition where all agents stay in place.

We conclude that the length of each segment is at most $O(n)$, since it comprises simple paths with possible waiting at the end of the segment, so that all agents finish their segments. Thus, the overall length of a minimal plan is $O(n \cdot r) = O(n^4)$.

Since verifying that a deocmposition is vertex-disjoint can be done in polynomial time in the length of the plan, and we have shown that if there is a plan, there is also one of polynomial length, we conclude that the problem is in **NP**, □

By combining Lemmas 3.2 and 3.3, we obtain the following theorem.

Theorem 3.4. *Problem 4 is **NP-Complete**.*

3.3 Explainability Versus Length

We now turn to study the tradeoff between plans that admit a small decomposition index and plans with small lengths. Traditionally, when solving the MAPF problem (Problem 1), one looks for plans that minimize the maximal time it takes an agent to reach its goal (i.e., length of the plan). As we now show, there is a potential tradeoff of the length and the minimal decomposition index of a plan.

Example 3.5. Consider the setting depicted in Figure 2. The lists of source and goal nodes are $s_1, \ldots, s_k$ and $g_1, \ldots, g_k$, respectively.

[2]The model in [28] allows the agents to stay in place, as we do. See Section 6 for the model where staying in place is not allowed.

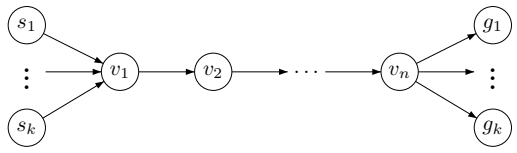


Figure 2: Setting for Example 3.5

Clearly, there is a (minimal-length) plan for all the agents to reach their goal within $O(n + k)$ time steps by traversing path $v_1, \ldots, v_n$ "back to back". However, the decomposition index of such a plan is $O(n + k)$, since the disjoint parts are of size 1 (i.e., at every step the paths intersect). However, if each agent waits for the previous one to reach its goal, the decomposition index is $O(k)$, but the length is $O(n \cdot k)$.

Remark 3.6. *One recent approach of tackling the MAPF problem is using* highways *[6], which are, roughly speaking, "preferable paths" that were successfully taken by other agents. Example 3.5 points to an intriguing aspect of disjoint decompositions: if agents want to follow the same path, they can only do so in separate segments of the decomposition. Thus, highways are intuitively bad for explainability. In Section 6.1, we discuss possible approaches to combine highways and disjoint decompositions.*

4 EXPLANATIONS OF CONTINUOUS MAPF

In the continuous setting, each agent moves in the environment according to the following dynamics:

$$\dot{x} = f(x, u), \quad x \in X \subseteq \mathbb{R}^n, \quad u \in U \subseteq \mathbb{R}^m \tag{1}$$

where X and U are the state and input spaces, respectively, and $f : X \times U \to X$ is an integrable and possibly nonlinear function[3].

Note that while we consider a two-dimensional environment $W \subseteq \mathbb{R}^2$ in this work, the state space of the robot is typically of a higher dimension, as it accounts for states such as velocity, heading angle, etc.

Traditionally, motion planning for continuous systems is studied for one agent with a start (source) and a goal state. The motion planning problem asks to compute a collision-free continuous trajectory in X from the start state to goal that respects the dynamics of the agent in (1). This problem is **NP-Hard** [9] in the best case and **undecidable** [3] in the worst case. To deal with such a difficult problem, especially, in a multi-agent setting with explanations, we take inspirations from formal control synthesis literature (e.g., [20, 22]) and approach the problem by abstracting the evolution of the agent in its state space to a discrete (finite) graph with simulation relation. Then, a discrete planner can be applied to this graph with the guarantee that the generated paths on the graph are correctly executable by the continuous dynamics.

To construct this abstraction graph $G = \langle V, E \rangle$, we first partition the environment into a set of finite regions, which induces a discretization in the state space X. We then associate each discrete region with a vertex in V of graph G. This discretization can be achieved using a grid, triangulation, voronoi tessellation etc. We

[3]For readability, we assume all agents have the same dynamics, but this assumption can be easily dropped.

call two regions *neighbors* if they share a facet. To enable transitions between neighbors $v_i, v_j \in V$, a motion plan that guarantees the trajectory does not exit $v_i \cup v_j$ while transitioning from v_i to v_j is required. Such a motion plan can be obtained using sampling-based (e.g., [21]) or control-based techniques. The advantage of sampling-based methods lies in their ease of handling complexity dynamics, but the obtained trajectories can have different durations for different pairs of neighbors. In the multi-agent setting, it gives rise to the issue of offset between continuous time and discrete clock of agents after each transition; hence, requiring synchronization through communication between agents. To avoid this problem, we use a control-based approach, namely control symbols [5].

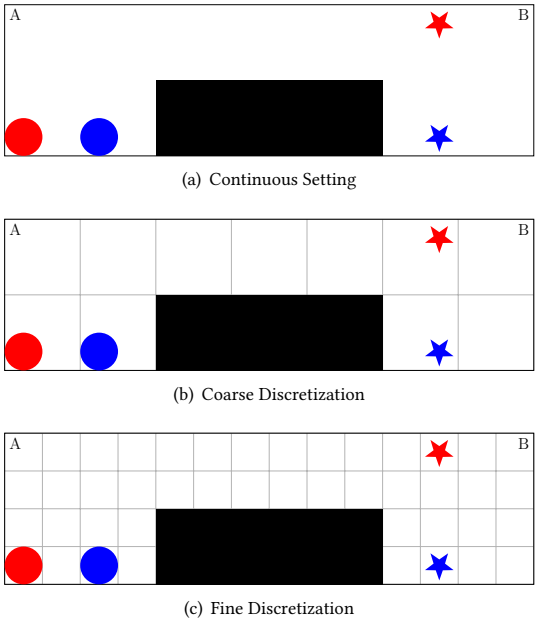
A control symbol $u_{ij} = (h_{ij}, \xi)$ consists of a feedback controller $h_{ij} : X \to U$, which is designed to drive the system from the center of v_i to the center v_j, and a termination rule (trigger) $\xi : \mathbb{R}_{\geq 0} \to \{0, 1\}$ that indicates when to terminate the execution of h_{ij}. We assume that h_{ij} is able to stabilize the system at the center of v_j, meaning that the system is able to remain at the center of v_j under controller h_{ij} after getting there. The construction of such a controller for various dynamical systems is detailed in [1]. If such a controller is feasible, then an edge directing node v_i to v_j is added to the graph G.

To avoid the issue of communication and enable automatic synchronization between agent actions, we design ξ as follows. Let Δt to be the maximum time duration needed for h_{ij} to drive the system to an ϵ-distance to the center of v_j for all neighbors $v_i, v_j \in V$. We design ξ to fire exactly when Δt is reached for all control symbols (i.e., $\xi = \delta(t - \Delta t)$, where δ is the Dirac function). With this design of control symbols u_{ij}, all agents are guaranteed to have completed their transits to their one-step target regions and synchronized by the time of switching to the next control symbol.

The constructed graph has a simulation relation with the continuous systems. That is, the continuous agents are able to simulate every behavior produced on graph G. The converse, however, is not true since the continuous agents can have behaviors that the abstraction G cannot simulate. Therefore, our explanation scheme is admissible for the technique above, in the sense that a disjoint decomposition in the discretization translates to a disjoint discretization in any continuous plan that implements the discrete one. Thus, intuitively, we can readily use our discrete methods of Section 3 in order to provide explanations for continuous MAPF.

A central challenge in the continuous setting is how to choose an appropriate discretization of the environment. Specifically, the size of the cells set a tradeoff: a finer discretization results in a larger graph, but allows more plans. This issue is still an open problem in motion planning and control, but it has significant effect on the explainability of plans, as we demonstrate below.

Example 4.1. Consider the environment depicted in Figure 3a. The two agents start in room A and need to reach room B. If the discretization is coarse, the corridor may be discretized as a single path of nodes (Figure 3b). Then, similarly to Example 3.5, the fastest plan is for the agents to take the corridor back to back, but this plan has a large decomposition index, whereas the plan with the minimal decomposition index, which is 2, requires one agent to wait in room A until the other has reached room B.



(a) Continuous Setting

(b) Coarse Discretization

(c) Fine Discretization

Figure 3: Two rooms connected by a corridor (the dark area is blocked). The circles and stars are the agents' start and goal locations, respectively.

However, if the discretization is finer, the corridor corresponds to two paths of vertices (Figure 3c), in which case there is a plan that is both short and has decomposition index 1, namely having both agents move through the corridor side by side.

Note that in fact, the fastest plan is obtained already in the coarse discretization, meaning that good explanability may actually come at the cost of not only giving up the fastest plan, but also refining the discretization after the fastest plan was found [4].

In Section 5, we discuss experimental results regarding the lengths of plans and the degree of refinement.

5 EXPERIMENTAL RESULTS

In this section, we present some experimental results. Our experiments are divided into two settings: discrete and continuous. In the discrete setting, we propose a fairly naive implementation of A^* to solve Problem 3, and discuss the practical challenges involved in the implementation. Then, using benchmarks taken from [30], we examine the tradeoff between explainability and length of plans.

In the continuous setting, we apply the methods suggested in Section 4 to discretize an environment and study the tradeoff of explanations, length of plans, and degree of refinement.

In the spirit of this work, our experiments are aimed at studying the notion of disjoint decompositions, rather than optimizing MAPF. Thus, our runtimes are always higher than state-of-the-art algorithms for MAPF.

[4] The scrupulous reader might notice that the example can be simplified by having only the corridor, without the rooms. We added the rooms to emphasize the corridor.

5.1 Finding Decompositions Using A*

One of the traditional approaches for solving MAPF (Problem 1) is search based, using A^* with various heuristics and optimizations [12, 16, 29]. The difficulty in search-based methods lies in the size of the state space, which is $|V|^k$, where k is the number of agents. Indeed, we must keep track of all the agents simultaneously to avoid collision. Notably, the state space grows exponentially in the number of agents, but only polynomially in the size of the graph.

Unfortunately, Problem 3 gives rise to another blow-up. Recall that within each segment of the decomposition, the paths taken by the agents are disjoint. Thus, in order to perform a search-based algorithm, each state must include, apart from the location of each agent i, the history of the states visited by Agent i in the current segment. Thus, each state encodes the current state of each agent i, as well as a set $S_i \subseteq V$. The overall size of the state space is thus $O(|V|^k \cdot 2^{|V| \cdot k})$, which is exponential both in the number of agents and the size of the graph. This can be slightly improved by noticing that the S_i are disjoint, and thus the subsets can be represented by a function $f : V \to \{1, \ldots, k, \perp\}$, so that $f(v)$ is the agent that had visited v in the current segment (where $f(v) = \perp$ means that v has not been visited yet). In this view, the state space is of size $O\left(|V|^k \cdot (k+1)^{|V|}\right)$, which is still exponential in both k and $|V|$.

The analysis above implies that even for 2 agents, applying A^* to Problem 3 is challenging. In Section 6, we discuss possible scaling solutions for search-based analysis, as well as other approaches.

5.2 Case Study: Discrete Domain

Our first case study concerns the discrete domain, where we examine the tradeoff between the length of plans and their minimal decomposition on standard benchmarks taken from [30]. Each benchmark scenario consists of a graph, and start and goal nodes for many agents. We examined two parameters for each scenario: the number of agents, and the maximum allowed decomposition index. Thus, for each scenario, we have tested an increasing number of agents, and a decreasing maximal decomposition index.

As described in Section 5.1, the exponential blowup both in the size of the graph and in the number of agents makes the algorithm prohibitively inefficient, with several instances timing out or running out of memory. Thus, our results are based on instances for which the algorithm terminated. Our code can be found in https://github.com/explainable-mapf/explainable-mapf.

In Table 1, we report our results. We observe two phenomena: first, in most cases the minimal index decomposition is not significantly longer than the shortest plan. This suggests that a smaller explanation is available without sacrificing much in length. Nonetheless, in some cases there is a significant increase in length, specifically in the "room-32-32-4-even-2" scenario. This typically occurs in the presence of corridors, that elicit behaviors as demonstrated in Example 3.5.

The second phenomenon is that the decomposition index of the shortest plans is not much higher than that of the minimal index plans. This is encouraging, as it means that often we can plan without explanations in mind, and compute the explanations after planning. As we proved in Section 3, the latter problem can be solved in polynomial time.

Scenario	ag	Shortest		Min. index		Time
		ind	Len	ind	Len	(sec)
empty-32-32-even-4	4	2	45	1	53	10.1
random-32-32-10-even-1	4	2	41	1	47	3.6
random-32-32-10-even-1	5	2	41	1	47	3.3
random-32-32-10-even-3	3	3	43	2	45	0.1
random-32-32-10-even-3	4	3	43	2	45	0.1
random-32-32-10-even-3	5	4	43	3	53	0.3
random-32-32-10-even-3	6	4	43	3	53	1.1
random-32-32-10-even-3	7	4	43	3	53	4.4
random-32-32-10-even-3	8	4	43	3	53	12.9
random-32-32-10-even-4	3	3	57	2	61	0.1
random-32-32-10-even-4	4	3	57	2	61	0.2
random-32-32-10-even-4	5	3	57	2	61	0.2
random-32-32-10-even-4	6	4	57	3	61	0.1
random-32-32-10-even-4	7	4	57	3	61	0.3
random-32-32-10-even-4	8	4	57	3	61	1.4
random-32-32-10-even-5	7	4	46	3	64	15.9
random-32-32-20-even-1	6	4	57	3	59	8.4
random-32-32-20-even-3	6	2	27	1	37	5.9
random-32-32-20-even-3	7	2	27	1	37	3.1
random-32-32-20-even-5	4	2	43	2	45	0.1
room-32-32-4-even-1	2	2	46	1	48	0.1
room-32-32-4-even-1	3	2	46	1	76	0.1
room-32-32-4-even-2	4	2	20	1	31	50.1
room-32-32-4-even-2	5	2	20	1	33	4.65

Table 1: Discrete benchmark results. The "scenario" column refers to the benchmarks in [30], "ag" states the number of agents. The "shortest" and "min index" columns refer to the shortest plan and the plan with minimal index found by our algorithm, where for each plan we report the index and the length.

5.3 Case Study: Continuous Domain

Next, we demonstrate our explanation scheme on agents with continuous dynamics. To show the strength of our abstraction technique, we assume agents with second-order unicycle dynamics given by

$$\dot{x} = v\cos\theta, \quad \dot{y} = v\sin\theta, \quad \dot{v} = u_1 \quad \dot{\theta} = u_2,$$

where x and y are the position, θ is the heading angle, and v is the speed of the agent. The control input u_1 and u_2 are the acceleration and turning rate. The considered environment is depicted in Figure 4.

We considered three disk-shaped agents, each with a source and goal region (shown as gray boxes and denoted by s_i and g_i for agent i in Figure 4). To discretize this environment, we used uniform grids of width 2, 1, and 0.5. If a cell of a grid overlaps with an obstacle, the whole cell is rendered obstacle. To complete the abstraction, we designed control symbols (h_{ij}, ξ) as follows. We first chose controllers $u_1 = \bar{u}_1\cos\theta + \bar{u}_2\sin\theta$ and $u_2 = \frac{\bar{u}_1\sin\theta + \bar{u}_2\cos\theta}{v}$, which lead to feedback linearization of the dynamics. Then, we used standard technique (i.e., LQR feedback control) to design $\bar{u}_1$ and $\bar{u}_2$ to stabilize the system around any point in space, satisfying

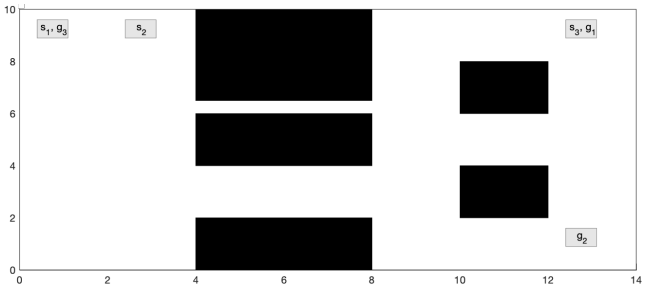


Figure 4: Environment for continuous case study. The source and goal regions for agent $i \in \{1, 2, 3\}$ are indicated by gray rectangles and denoted by s_i and g_i, respectively.

the requirement on h_{ij}. The trigger ξ is the maximum time duration for h_{ij} to drive the system between all v_i and v_j neighbors.

The results are shown in Table 2, and the continuous plans that correspond to the minimal decomposition plans are given in Figures 5–7.

Grid width	Min. length		Min. index	
	ind	Len	ind	Len
2	7	38	3	40
1	4	30	2	31
0.5	8	23.5	2	25.5

Table 2: Continuous case study results. The "width" column refers to the width of the discretization. The remaining columns are similar to Table 1. Note that the reported lengths are in continuous length units, rather than discrete steps.

As can be seen from Table 2, the minimal index plans are not significantly longer than the shortest plans. However, there is a significant gain in the decomposition index. Moreover, we see that the degree of refinement does effect the decomposition: the coarse discretization achieved longer plans and incurred a higher decomposition index.

A more detailed examination of the setting shows that while the width 1 discretization already allows for a decomposition of index 2 (Figure 6), the finer 0.5 discretization allows the agents to take the smaller corridor, and hence achieve shorter plans. This, however, incurs a significant increase in the decomposition index, since we get the behavior of Example 4.1. Nonetheless, by utilizing both corridors, we can achieve a minimal decomposition with a relatively short plan (see Figure 7).

We conclude that, as expected, refining the discretization allows for shorter plans, and smaller decompositions. Surprisingly, there is hardly any lengthening in plans when searching for minimal decompositions. This suggests that it is worthwhile to plan for minimal decompositions while discretizing to attain short plans.

6 DISCUSSION AND FUTURE WORK

In this work we have laid down principles for devising an explanation scheme for algorithms. We demonstrate these principles in the context of MAPF, where we devise an explanation scheme that is sound, and offers a tradeoff between completeness and simplicity.

We have shown that merely explaining the result of existing MAPF algorithms is tractable, and in fact can be done greedily. This implies that our scheme can be readily incorporated in existing frameworks. Moreover, our experimental results show that the decomposition index for shortest plans is, in practice, often not significantly higher than the minimal decomposition index.

We have shown that the "deeper" problem of finding plans that admit a short explanation is **NP-Complete**, and that solving it using search-based methods introduces a blowup in the state space that is not present in traditional MAPF.

In addition, we showed that there is a potential tradeoff between the length of a plan and the length of an explanation for it, meaning that longer plans sometimes have shorter explanations, and vice-versa. Finally, we also demonstrated that the problem of MAPF with explanations in the continuous setting is very hard. Through means of abstraction, it is possible to overcome local motion planning and synchronization issues and employ discrete planners. However, the choice of discretization, which is non-trivial, has some effect on the explanations and plan lengths.

6.1 Future Work

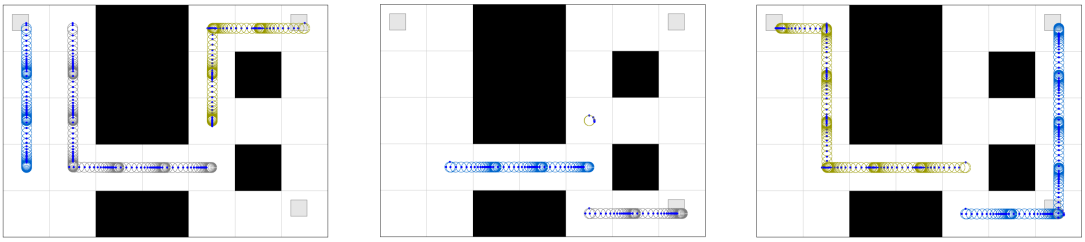
We now turn to discuss some future research directions concerning explainability.

Optimization. Since the focus of this work is the introduction of an explanation scheme for MAPF, we have used naive implementations in our experiments. However, state-of-the-art algorithms offer many optimizations that might be adaptable to output explanations as well. As we have observed in 3.6, the highway method of [6] stands somewhat in contrast with our notion of disjoint decomposition. One possible combination of highways and explanations could rely on identifying "wide" highways, that allow agents to travel in parallel. These admit shorter explanation, and also allow for shorter plans.

Another optimization technique that may prove useful is the following simple observation: a plan has a decomposition with m segments, if it can be broken into m disjoint plans. Thus, one approach to find a plan with a decomposition of index m is to find $m-1$ configurations for the agents, such that there are disjoint plans from one configuration to the next. Then, we can use algorithms for finding disjoint paths. While the latter problem is still **NP-Complete**, there are some works concerning it [8, 14, 27], which may shed new light on MAPF.

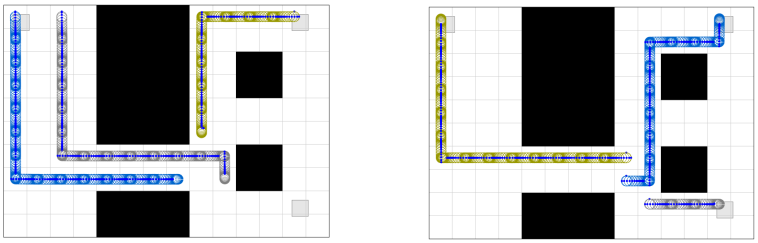
A potentially promising approach is to formulate MAPF with explanations as a SAT problem. SAT/SMT solvers have proved to have some success in handling MAPF [31, 32, 34, 36]. Preliminary experiments we conducted show that a naive encoding of Problem 3 performs very badly with SAT solvers. However, it is possible that using state-of-the-art techniques for solving MAPF with SAT/SMT solvers can also work in our setting.

Sampling-based Planners. In our work on the continuous settings, we based our approach on an abstraction technique, which lifts the problem to the discrete domain. This greatly simplifies the problem. However, as discussed in [21], the advantages of such



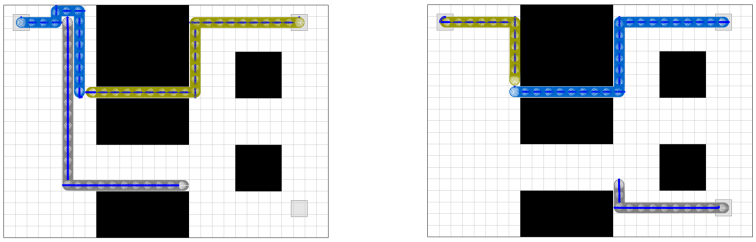
(a) Part 1 (b) Part 2 (c) Part 3

Figure 5: Decomposition plans for a discretization of width 2.



(a) Part 1 (b) Part 2

Figure 6: Decomposition plans for a discretization of width 1.



(a) Part 1 (b) Part 2

Figure 7: Decomposition plans for a discretization of width 0.5.

methods come at the cost of sacrificing completeness since these abstractions typically hold only a simulation relation with the continuous system, rather than bi-simulation. In addition, the problem of discretization and its effects on explanations and plan lengths is embedded in such approaches. An interesting future direction is to formulate Problem 3 directly in the continuous domain. In this setting, the explanations will require the projection of the agents' trajectory segments onto the environment to be non-intersecting. Given recent performance boosts in motion planning for multi-agent systems through sampling-based techniques, a promising direction would be to explore these methods to generate plans with explanations in the continuous domain as an alternative to the discretization and abstraction approaches.

Disjoint Decompositions in 3D. Our work focuses on the two dimensional domain, where paths can be depicted in pictures. In the three dimensional setting, a new challenge arises, which is to display the disjoint paths. One naive way of solving this is to split the three dimensional space into layers, and treat each layer as a two dimensional space. This approach is reasonable in e.g. air traffic control, where flying space is already treated in this fashion.

However, in settings such as UAVs, this is less practical. Therefore, we suggest the following extension to our explanation scheme: instead of outputting only disjoint segments, we output for each segment also a *viewpoint*, from which the projection of this segment is disjoint in the two-dimensional sense. Thus, an explanation comprises disjoint segments, with a possibly different viewpoint of each segment. Computing such viewpoints is a challenge that is out of the scope of this work, and is thus left for future research.

REFERENCES

[1] Ali-Akbar Agha-Mohammadi, Suman Chakravorty, and Nancy M Amato. 2014. FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research* 33, 2

(2014), 268–304.

[2] Shaull Almagor, Dmitry Chistikov, Joël Ouaknine, and James Worrell. 2018. O-Minimal Invariants for Linear Loops. In *45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[3] Rajeev Alur, Costas Courcoubetis, Nicolas Halbwachs, Thomas A Henzinger, P-H Ho, Xavier Nicollin, Alfredo Olivero, Joseph Sifakis, and Sergio Yovine. 1995. The algorithmic analysis of hybrid systems. *Theoretical computer science* 138, 1 (1995), 3–34.

[4] R. Bartak, J. Svancara, and M. Vlk. 2018. A Scheduling-Based Approach to Multi-Agent Path Finding with Weighted and Capacitated Arcs. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 748–756.

[5] RW Brockett. 1990. Formal languages for motion description and map making. *Robotics* 41 (1990), 181–191.

[6] Liron Cohen and Sven Koenig. 2016. Bounded Suboptimal Multi-Agent Path Finding Using Highways.. In *IJCAI*. 3978–3979.

[7] L. Cohen, S. Koenig, S. Kumar, G. Wagner, H. Choset, D. Chan, and N. Sturtevant. 2018. Rapid Randomized Restarts for Multi-Agent Path Finding: Preliminary Results. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 1909–1911.

[8] Marek Cygan, Dániel Marx, Marcin Pilipczuk, and Michal Pilipczuk. 2013. The planar directed k-vertex-disjoint paths problem is fixed-parameter tractable. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*. IEEE, 197–206.

[9] Bruce Donald, Patrick Xavier, John Canny, John Canny, John Reif, and John Reif. 1993. Kinodynamic motion planning. *Journal of the ACM (JACM)* 40, 5 (1993), 1048–1066.

[10] Filip Karlo Došilović, Mario Brčić, and Nikica Hlupić. 2018. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*. IEEE, 0210–0215.

[11] Rebecca Eifler, Michael Cashmore, Jörg Hoffmann, Daniele Magazzeni, and Marcel Steinmetz. 2019. Explaining the space of plans through plan-property dependencies. In *Proceedings of the 2nd Workshop on Explainable Planning (XAIP 2019)*.

[12] Ariel Felner, Meir Goldenberg, Guni Sharon, Roni Stern, Tal Beja, Nathan Sturtevant, Jonathan Schaeffer, and Robert Holte. 2012. Partial-expansion A* with selective node generation. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*.

[13] A. Felner, R. Stern, E. Shimony, M. Goldenberg, G. Sharon, N. Sturtevant, G. Wagner, and P. Surynek. 2017. Search-Based Optimal Solvers for the Multi-Agent Pathfinding Problem: Summary and Challenges. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*. 28–37.

[14] Steven Fortune, John Hopcroft, and James Wyllie. 1980. The directed subgraph homeomorphism problem. *Theoretical Computer Science* 10, 2 (1980), 111–121.

[15] Maria Fox, Derek Long, and Daniele Magazzeni. 2017. Explainable Planning. In *Explainable planning*. International Joint Conferences on Artificial Intelligence.

[16] Meir Goldenberg, Ariel Felner, Roni Stern, Guni Sharon, and Jonathan Schaeffer. 2012. A* variants for optimal multi-agent pathfinding. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

[17] D Gunning. 2016. *Explainable Artificial Intelligence (XAI) DARPA-BAA-16-53*. Technical Report. Technical report, Defense Advanced Research Projects Agency (DARPA).

[18] David H Hubel and Torsten N Wiesel. 1959. Receptive fields of single neurones in the cat's striate cortex. *The Journal of physiology* 148, 3 (1959), 574–591.

[19] Subbarao Kambhampati. 2019. Synthesizing explainable behavior for human-AI collaboration. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1–2.

[20] Marius Kloetzer and Calin Belta. 2008. A fully automated framework for control of linear systems from temporal logic specifications. *IEEE Trans. Automat. Control* 53, 1 (2008), 287–297.

[21] Athanasios Krontiris, Qandeel Sajid, and Kostas E Bekris. 2012. Towards using discrete multiagent pathfinding to address continuous problems. In *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*.

[22] Morteza Lahijanian, M. Kloetzer, S. Itani, C. Belta, and S.B. Andersson. 2009. Automatic deployment of autonomous cars in a robotic urban-like environment (RULE). In *In Proceedings of the 2009 IEEE International Conference on Robotics and Automation*. Kobe, Japan, 2055–2060.

[23] Sebastian Lapuschkin, Stephan Wäldchen, Alexander Binder, Grégoire Montavon, Wojciech Samek, and Klaus-Robert Müller. 2019. Unmasking Clever Hans predictors and assessing what machines really learn. *Nature communications* 10, 1 (2019), 1096.

[24] H. Ma, D. Harabor, P. Stuckey, J. Li, and S. Koenig. 2019. Searching with Consistent Prioritization for Multi-Agent Path Finding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*. (in print).

[25] Matthias Middendorf and Frank Pfeiffer. 1993. On the complexity of the disjoint paths problem. *Combinatorica* 13, 1 (1993), 97–107.

[26] Bruce A Reed, Neil Robertson, Alexander Schrijver, and Paul D Seymour. 1993. Finding disjoint trees in planar graphs in linear time. *Contemp. Math.* 147 (1993), 295–295.

[27] Alexander Schrijver. 1994. Finding k disjoint paths in a directed planar graph. *SIAM J. Comput.* 23, 4 (1994), 780–788.

[28] Daniel Kornhauser Gary Miller Paul Spiralris. 1984. COORDINATING PEBBLE MOTION O_N GRAPHS, THE DIAMETER OF PERMUTATION GROUPS, AND APPLICATIONS. (1984).

[29] Trevor Scott Standley. 2010. Finding optimal solutions to cooperative pathfinding problems. In *Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[30] Roni Stern, Nathan R. Sturtevant, Dor Atzmon, Thayne Walker, Jiaoyang Li, Liron Cohen, Hang Ma, T. K. Satish Kumar, Ariel Felner, and Sven Koenig. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. *Symposium on Combinatorial Search (SoCS)* (2019), 151–158.

[31] Pavel Surynek. 2012. Towards optimal cooperative path planning in hard setups through satisfiability solving. In *Pacific Rim International Conference on Artificial Intelligence*. Springer, 564–576.

[32] Pavel Surynek, Ariel Felner, Roni Stern, and Eli Boyarski. 2016. Boolean satisfiability approach to optimal multi-agent path finding under the sum of costs objective. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1435–1436.

[33] P. Surynek, A. Felner, R. Stern, and E. Boyarski. 2016. An Empirical Comparison of the Hardness of Multi-Agent Path Finding under the Makespan and the Sum of Costs Objectives. In *Proceedings of the Symposium on Combinatorial Search (SoCS)*. 145–147.

[34] Pavel Surynek, Jiří Švancara, Ariel Felner, and Eli Boyarski. 2017. Integration of independence detection into sat-based optimal multi-agent path finding-A novel sat-based optimal MAPF solver. In *International Conference on Agents and Artificial Intelligence*, Vol. 2. SCITEPRESS, 85–95.

[35] Shiming Tang, Tai Sing Lee, Ming Li, Yimeng Zhang, Yue Xu, Fang Liu, Benjamin Teo, and Hongfei Jiang. 2018. Complex pattern selectivity in macaque primary visual cortex revealed by large-scale two-photon imaging. *Current Biology* 28, 1 (2018), 38–48.

[36] Jiangxing Wang, Jiaoyang Li, Hang Ma, Sven Koenig, and TK Kumar. 2019. A New Constraint Satisfaction Perspective on Multi-Agent Path Finding: Preliminary Results. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2253–2255.