# Generating Safe Policies for Multi-Agent Path Finding with Temporal Uncertainty

Jiří Švancara[1][a], David Zahrádka[2,3][b], Mrinalini Subramanian[1][c],
Roman Barták[1][d] and Miroslav Kulich[3][e]

[1]*Faculty of Mathematics and Physics, Charles University, Prague, Czech Republic*
[2]*Dept. of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic*
[3]*Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, Prague, Czech Republic*
{svancara, subramanian, bartak}@ktiml.mff.cuni.cz, {david.zahradka, miroslav.kulich}@cvut.cz

Abstract: Multi-Agent path finding (MAPF) deals with the problem of finding collision-free paths for a group of mobile agents moving in a shared environment. In practice, the duration of individual move actions may not be exact but rather spans in a given range. Such extension of the MAPF problem is called MAPF with Temporal Uncertainty (MAPF-TU). In this paper, we propose a compilation-based approach to generate safe agents' policies solving the MAPF-TU problem. The policy guarantees that each agent reaches its destination without collision with other agents provided that all agents move within their predefined temporal uncertainty range. We show both theoretically and empirically that using policies rather than plans is guaranteed to solve more types of instances and find a better solution.

1 INTRODUCTION

Coordinating a fleet of moving agents is an important problem with practical applications such as warehousing (Wurman et al., 2008), airplane taxiing (Morris et al., 2015), or traffic junctions (Dresner and Stone, 2008). *Multi-Agent Path Finding* (MAPF) is an abstract model of this coordination problem, where we are looking for collision-free paths for agents moving in a shared environment represented by a graph. The problem of finding an optimal MAPF solution has been shown to be NP-hard for a wide range of cost objectives (Surynek, 2010).

A practical extension of the classical MAPF problem is *MAPF with Temporal Uncertainty* (MAPF-TU) (Shahar et al., 2021), where each transition between two locations is associated with a lower and upper bound on the time it takes any agent to move. This extension models the real world, where the movement of agents may be affected by terrain, imperfect execution, uncertainty in localization, or unexpected obsta-

cles. In the original paper (Shahar et al., 2021), the authors defined the problem and presented an optimal algorithm based on the Conflict-Based Search (CBS) algorithm (Sharon et al., 2015). Specifically, they assumed blind execution of the found plan and required that the plan be safe (i.e. collision-free) for any duration of the traversal.

In this paper, we extend the concept of MAPF-TU to include different actions depending on the traversal time, thus, creating a policy. This extension allows the agent to move more efficiently and possibly reach the goal faster, yet still guarantees safety. A reduction-based model is introduced that facilitates policy creation. Using a reduction-based approach is quite natural as the variables modeling the policy are already present in the model for classical MAPF. Example instances are provided to demonstrate the benefit of using policies instead of plans. Also, theoretical guarantees on the solution's existence and the length of the solution are provided. Lastly, empirical experiments are carried out to compare the benefits of policies in terms of the length of the final plan.

[a] https://orcid.org/0000-0002-6275-6773
[b] https://orcid.org/0000-0002-7380-8495
[c] https://orcid.org/0009-0002-9511-9217
[d] https://orcid.org/0000-0002-6717-8175
[e] https://orcid.org/0000-0002-0997-5889

2 RELATED WORK

Conflict Based Search (CBS), a search-based solver, is a complete and optimal MAPF solver designed to solve MAPF problems by iteratively planning for each agent individually on the lower level. A top-level search over a constraint tree is performed to identify conflicts between the single-agent paths and resolve them (Sharon et al., 2015).

A reduction-based approach to solve MAPF is to create variables modeling the possible location of each agent and generating a formula in the given formalism to enforce the allowed movement and avoid any conflicts. The reduction-based solver may make use of, for example, Boolean satisfiability (SAT) (Surynek, 2012), Answer Set Programming (ASP) (Erdem et al., 2013), or Constraint Satisfaction Problem (CSP) (Ryan, 2010).

Various extensions to MAPF have been proposed to address uncertainties during execution.

The notion of k-robustness has been proposed (Atzmon et al., 2018) to produce plans that are safe even if any agent gets delayed up to k timesteps anywhere along its path. Such an approach, however, does not take into account that delays are more or less likely to occur at different locations on the map and may produce an overly cautious plan.

A post-processing algorithm MAPF-POST (Hönig et al., 2016) refines a valid MAPF plan to mitigate risks while executing on robots with varying velocity constraints. Similarly, the Action Dependency Graph (ADG) (Hönig et al., 2019) utilizes a precedence relation on actions. During execution, action can be performed only after the previous action has been safely finished. This ensures safe execution; however, it may introduce unnecessary delays.

Another extension is MAPF with obstacle uncertainty (Shofer et al., 2023), where some vertices are blocked by obstacles that are initially unknown to the agents. Agents can sense whether these positions are traversable only when they are reached. One of the proposed solutions is to create plan trees where the correct branch of the tree is selected based on the obstacle observation.

3 PROBLEM DEFINITION AND PROPERTIES

We base our setting on commonly used formal definitions from literature. Namely, *classical multi-agent pathfinding* (Stern et al., 2019) and *multi-agent pathfinding with temporal uncertainty* (Shahar et al., 2021). For the latter problem, we define a different task and solution than was previously used. We compare these approaches theoretically and on examples.

3.1 Definitions

An instance of *classical multi-agent pathfinding (MAPF)* is a tuple $\mathcal{M} = (G, A)$, where $G = (V, E)$ is a graph representing the shared environment and A is a set of n agents. Each agent $a_i \in A$ is represented by its start and goal location s_i and g_i, respectively. The time is considered discrete and at each timestep, each agent can move to a neighboring location (i.e. move over an edge $(u, v) \in E$), or wait in its current location.

The task is to find a *single-agent plan* τ_i [1] for each agent a_i. A plan is a sequence of actions in the form of pairs $(u, v) \in E$, representing that an agent is moving over an edge. Note that an agent can wait in any vertex; thus, loops (v, v) are allowed for each vertex. We will use the following notation to represent the t-th action $- \tau_i[t] = (u, v)$ meaning that after performing t actions, agent a_i is located in vertex v, and $|\tau_i| = k$ meaning that the plan for agent a_i consists of k actions.

The solution to classical MAPF is a set of plans τ_i for each agent a_i and is said to be valid if each plan τ_i navigates the corresponding agent a_i from its initial location s_i to its goal location g_i, specifically, $\tau_i = ((v_0, v_1), (v_1, v_2), \ldots (v_{k-1}, v_k))$, where $v_0 = s_i$ and $v_k = g_i$, no two agents occupy the same vertex at the same time (i.e. no vertex conflict), and no two agents traverse the same edge at the same time in either direction (i.e. no swapping conflict).

An instance of *multi-agent pathfinding with temporal uncertainty (MAPF-TU)* is a tuple $\mathcal{M}^{TU} = (G^w, A)$, where $G^w = (V, E, w^-, w^+)$ is a graph representing a shared environment and A is a set of n agents. The graph contains in addition functions $w^- : E \to \mathbb{N}$ and $w^+ : E \to \mathbb{N}$ returning the minimal and maximal duration it takes an agent to traverse a given edge, respectively, and $\forall (u, v) \in E : w^-((u, v)) \leq w^+((u, v))$. The set of agents is identical to the classical MAPF instance.

The task is to find a *single-agent policy* π_i for each agent a_i. A policy is a function $\pi_i : V \times \{0, \ldots T\} \to E$ mapping possible states (i.e. location in time) to actions, where T is some sufficiently large bound on the number of timesteps. A state is the agent's location at a given time and an action is in the form of pairs $(u, v) \in E$, representing that an agent is moving over an edge. Again, an agent can wait in any vertex, thus loops (v, v) are allowed for each vertex

---

[1] In the literature, the plan is usually denoted as π, however, we will reserve π for policy and use τ for plans instead.

and are assumed to have no temporal uncertainty (i.e. $w^-((v,v)) = 1$ and $w^+((v,v)) = 1$). We will use the following notation – $\pi_i[u,t] = (u,v)$ meaning that if an agent a_i is located in vertex u in timestep t, it will move through edge (u,v).

The solution to MAPF-TU is a set of policies π_i for each agent a_i and is said to be valid if each policy π_i navigates the corresponding agent a_i from its initial location s_i to its goal location g_i. Specifically, $\pi_i[s_1,0]$ is defined. If there exists $\pi_i[u,t] = (u,v)$ then $\pi_i[v,t+w]$, where $w \in \{w^-((u,v)), w^+((u,v))\}$ is the possible duration of transition of edge (u,v), is also defined. Lastly, there exists $\pi_i[u,t] = (u,g_i)$ for some edge (u,g_i) such that $t + w^+((u,g_i)) \leq T$. Furthermore, there are no potential conflicts. Specifically, to forbid vertex conflicts, no two single-agent policies can navigate the agents into the same location in intersecting time intervals – $\{t_1 + w^-((u_1,v)), t_1 + w^+((u_1,v))\} \cap \{t_2 + w^-((u_2,v)), t_2 + w^+((u_2,v))\} = \emptyset$ for $\pi_i[u_1,t_1] = (u_1,v)$ and $\pi_j[u_2,t_2] = (u_2,v)$. To forbid edge conflicts, no two single-agent policies can navigate the agents over the same edge in intersecting time intervals – $\{t_1 + w^-((u,v)), t_1 + w^+((u,v))\} \cap \{t_2 + w^-((u,v)), t_2 + w^+((u,v))\} = \emptyset$ for $\pi_i[u,t_1] = (u,v)$ and $\pi_j[u,t_2] = (u,v)$. Similarly, to forbid swapping conflicts, no two single-agent policies can navigate the agents over the same edge in the opposite direction in intersecting time intervals – $\{t_1 + w^-((v,u)), t_1 + w^+((v,u))\} \cap \{t_2 + w^-((u,v)), t_2 + w^+((u,v))\} = \emptyset$ for $\pi_i[v,t_1] = (v,u)$ and $\pi_j[u,t_2] = (u,v)$.

Note that edge conflicts need to be forbidden explicitly as the traversal duration can be non-unit. In the classical MAPF with unit duration, the edge conflict is forbidden implicitly by forbidding vertex conflicts.

3.2 Plan and Policy Comparison

In the original paper defining this problem (Shahar et al., 2021), the authors define a solution to MAPF-TU as a set of plans for each agent, similar to the classical MAPF. However, this means that the agent always performs the same action no matter the actual duration it took to traverse an edge. This decision comes from the assumption that the agents cannot sense their location, cannot measure time, cannot sense other agents, and cannot communicate. These assumptions may be too strict in practice. Therefore, we will assume that the agent can sense its location and measure time; as a result, the agent knows the time it took to traverse the edge and may choose a different action (i.e. make use of policy). Note that the agents still do not communicate; therefore, the found

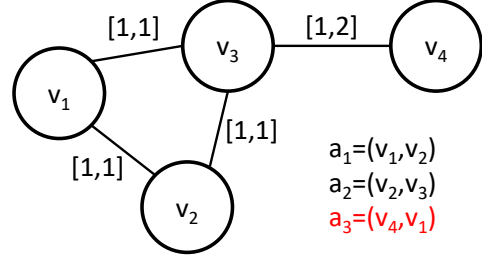policy has to be safe for any traversal time of the other agents.

Figure 1: An example instance that can not be solved by a plan but a policy ensures safe execution. The red agent a_3 needs to move over an edge with temporal uncertainty (the move may take 1 or 2 timesteps), then all three agents need to move over the cycle counter-clockwise. For a plan-based solution, the movement over the cycle is never safe, as we do not know when the agent a_3 reached v_3. A policy ensures that the agent a_3 waits for one timestep if it traverses the edge in a single timestep and then all agents move.

Proposition 1. *If there exists a classical MAPF solution, there also exists a policy-based solution to MAPF-TU with any positive w^- and w^+.*

Proof sketch. Any classical MAPF plan may be extended to a policy by adding an extra wait action to ensure the agents are synchronized for any traversal duration. It can be shown for each possible MAPF conflict that such synchronization is possible. □

The Proposition 1 does not hold true for the plan-based solution. See Figure 1 for example introduced in the original MAPF-TU paper (Shahar et al., 2021). The only policy to solve the problem is for the red agent to wait in case it transitioned the edge in a single timestep. In timestep 2, the red agent can enter the cycle. Lastly, all agents can rotate over the cycle to reach their goal. There is no plan-based solution, as it is not clear when it is safe for the agents to start rotating over the cycle.

There are two possible cost functions to be optimized in MAPF-TU (Shahar et al., 2021) – *optimistic sum of costs* (optimistic soc) and *pessimistic sum of costs* (pessimistic soc) – the sum of the earliest possible times each agent is in its goal and the sum of the latest possible times each agent is in its goal, respectively. It has been shown that optimizing either of those cost functions yields different solutions.

Proposition 2. *A policy-based solution always produces a solution with equal or better cost compared to a plan-based solution when optimizing optimistic soc or pessimistic soc.*

Proof. The found policy can be identical to the optimal plan; thus, it is never worse. On the other hand,

there are instances where the policy reduces the optimal cost. See Figure 2 for an example. □
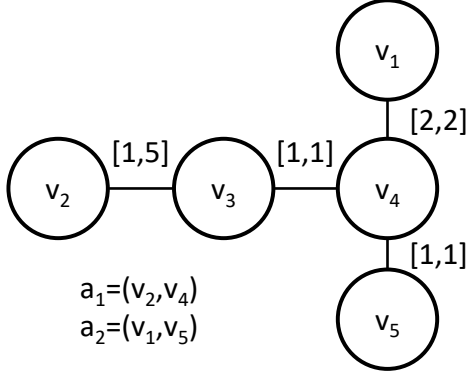

Figure 2: An example where a policy-based approach can find a better solution than a plan-based approach. Agent a_2 needs to move without any waiting actions and will reach its goal after 3 timesteps. Using a plan, agent a_1 needs to wait for 1 timestep to make sure that its goal is vacant. Using a policy, agent a_1 waits only if the first transition took exactly 1 timestep, otherwise, it is safe to move to its goal. Thus, the policy-based solution produces a solution with the pessimistic sum of costs being 1 less than the plan-based solution.

4 REDUCTION-BASED MODEL

To find a policy for the MAPF-TU problem, we leverage an SAT-based solver and extend an encoding for a classical MAPF problem from (Barták and Svancara, 2019). To model the classical MAPF, variables $At(v,a,t)$ are used to represent agents' locations, and variables $Pass((u,v),a,t)$ are used to represent agents' movement over an edge, i.e. performing an action. These variables exist for all possible reachable positions, and the encoding forces the SAT solver to choose just one position for each agent, yielding a valid plan. To find a policy, we use the same variables; however, we let the solver choose several different possible positions for each agent and for each position one valid action, thus yielding a valid policy.

Assume that there is some bound on the number of allowed timesteps T. To model the policy, we use the following constraints. Note that the constraints are written as (in)equalities and variables are assumed to have domains of $\{0,1\}$, which is easy to translate to SAT (Zhou and Kjellerstrand, 2016). We present the constraints in this fashion to improve readability and interpretability.

$$\forall a_i \in A : At(s_i,a_i,0) = 1 \tag{1}$$
$$\forall a_i \in A : At(g_i,a_i,T) = 1 \tag{2}$$

$$\forall v \in V, \forall a_i \in A, \forall t \in \{0,\ldots,T-1\} :$$
$$At(v,a_i,t) \implies \sum_{(v,u) \in E} Pass((v,u),a,t) = 1 \tag{3}$$

$$\forall (v,u) \in E, \forall a_i \in A, \forall t \in \{0,\ldots,T-1\} :$$
$$Pass((v,u),a,t) \implies At(v,a_i,t) \tag{4}$$

$$\forall (v,u) \in E, \forall a_i \in A,$$
$$\forall t \in \{0,\ldots,T-w^+((v,u))\},$$
$$\forall w \in \{w^-((v,u)),w^+((v,u))\} :$$
$$Pass((v,u),a,t) \implies At(u,a,t+w) \tag{5}$$

$$\forall v \in V, \forall t \in \{0,\ldots,T\} : \sum_{a_i \in A} At(v,a_i,t) \leq 1 \tag{6}$$

$$\forall (u,v) \in E : u \neq v, \forall t \in \{0,\ldots,T-1\} :$$
$$\sum_{\substack{a_i \in A, \\ (x,y) \in \{(u,v),(v,u)\}, \\ w \in \{w^-((x,y)),w^+((x,y))\}}} (Pass((x,y),a_i,t+w) \leq 1 \tag{7}$$

Constraints 1 and 2 model the start and goal locations of each agent. Constraints 3–5 model a correct movement of each agent. Specifically, 3 ensures that if an agent is present in a location, it chooses exactly one outgoing edge (i.e. action). Constraint 4 ensures that if an agent is using an edge, it must have been in the corresponding vertex at the correct time. Lastly, 5 ensures that if an agent is moving over an edge, it will arrive at the connected vertex at the next timestep. In fact, it will arrive in all the possible next timesteps based on the temporal uncertainty of the edge. This constraint allows the agent to "duplicate" itself; therefore, the policy is finding actions for all possible positions of the agent. Note that we do not allow an agent to move over an edge if the uncertainty allows the agent to arrive after the global time limit T. Constraints 6 forbid vertex conflicts, while constraint 7 forbids swapping and edge conflicts. Forbidding the edge and swapping conflict is more technical than in the classical MAPF model since the edges have non-unit lengths and are associated with time uncertainty. Intuitively, 7 forbids using an edge in any direction and in any overlapping timesteps.

Iteratively increasing T until a solvable formula is generated guarantees finding a pessimistic makespan optimal solution. To optimize the pessimistic sum of costs, a numerical constraint is introduced stating that at most k extra actions may be used (Surynek et al., 2016). An extra action refers to an action that is performed after the timestep D_i by agent a_i, where D_i is the distance from s_i to g_i assuming the edges have length dictated by w^+ (i.e. the pessimistic shortest path). We iteratively increase k and T by one until a solvable formula is created.

5 EXPERIMENTS

5.1 Instance Setup

The experiments are conducted on different grid map types (empty map and map with randomly placed obstacles), varying sizes (8 by 8, 16 by 16, and 24 by 24), uncertainty levels ($U \in \{1,2,3\}$, where for each edge, w^- is selected randomly from $\{1,\ldots U\}$ and w^+ is selected randomly from $\{w^-,\ldots w^- + U\}$), and number of agents (from 2 to 20 agents with an increment of 2). For each setting, we created 5 instances with randomly placed starting and goal locations. Together, we created 900 individual instances.

The experiments were conducted on a computer with Intel® Core™ i5-6600 CPU @ 3.30GHz × 4 and 64GB of RAM. The SAT-based solver is implemented using the Picat language (Picat language and compiler version 3.7) (Zhou and Kjellerstrand, 2016). As a comparison, we use the CBS-TU algorithm producing pessimistic optimal plans introduced in (Shahar et al., 2021). Each solver was given a 300s time limit per instance. The code and all the results can be found at https://github.com/svancaj/MAPF-TU.

5.2 Result

Table 1: Number of solved instances by the CBS-TU algorithm and by the SAT-based solver. The results are split by the uncertainty and the number of agents.

agents	CBS-TU			SAT-based		
	U=1	U=3	U=5	U=1	U=3	U=5
2	30	30	30	30	30	30
4	30	30	30	30	30	30
6	30	29	29	30	30	30
8	30	24	21	30	30	30
10	29	18	9	30	30	28
12	26	10	7	30	28	25
14	20	5	3	30	27	17
16	13	0	2	30	20	12
18	10	0	1	24	12	3
20	8	0	0	21	9	0

The number of solved instances by each approach is reported in Table 1. We can see that as the number of agents and the uncertainty increases, the problem becomes harder to solve for both CBS-TU and the SAT-based solver. In the experiments, we are mostly interested in showing the possible improvement to the solution cost by using policies rather than showing the computational efficiency of the proposed method. As was shown, both search-based and reduction-based approaches excel at different types of maps (Svancara et al., 2024).

Table 2 shows the measured quality of the found plans by CBS-TU and the policies found by our SAT-based solver. The optimized and reported cost function is the pessimistic sum of costs. The results indicate that as the number of agents increases, the sum of costs also increases. This is indeed to be expected, as each agent contributes to the total cost. Similarly, as the uncertainty increases, the total sum of costs increases. Again, this is to be expected, as increased uncertainty means that each traversal may take longer. However, the increase in the sum of costs does not scale with the same factor as the uncertainty level, as the agents prefer to traverse edges with shorter traversal time.

Table 2: The sum of costs of the pessimistic optimal plans found by the CBS-TU algorithm and the pessimistic optimal policies found by the SAT-based solver. The results are split by the uncertainty and the number of agents. Other parameters of the instances are averaged.

agents	CBS-TU			SAT-based		
	U=1	U=3	U=5	U=1	U=3	U=5
2	30,63	53,13	75,10	30,63	53,03	74,87
4	62,40	107,23	151,70	62,30	106,33	150,47
6	97,67	169,21	240,76	97,40	166,00	235,23
8	132,77	220,83	332,19	132,07	224,77	319,23
10	167,48	286,06	392,22	166,10	283,77	386,75
12	208,77	367,30	509,57	200,10	332,82	451,56
14	249,15	414,20	649,67	232,43	389,85	495,06
16	316,31	-	723,50	269,63	446,90	580,67
18	352,90	-	715,00	313,63	514,75	744,00
20	386,38	-	-	343,71	500,78	-

Table 2 does not show the improvement of policies over plans, as the numbers are skewed by the fact that the SAT-based solver solved more instances than CBS-TU (the number of solved instances can be seen in Table 1). For example, for 18 agent and $U = 5$, the average solution cost for CBS-TU is 715, while for the SAT-based solver it is 744, which does not correspond to Proposition 2. This is caused by the fact that CBS-TU managed to solve only one instance with a cost of 715, while the SAT-based solver solved 3 instances with costs of 705, 645, and 882. The first one being the same instance CBS-TU also solved.

A more representative result showing the improvement can be seen in Table 3. We calculate the cost of the solution over the lower bound for both solvers for instances that were solved by both solvers. I.e. the cost of the solution consists of a lower bound and some δ. As the lower bound cannot be improved, the only improvement to the solution can be done by finding a lower δ. The reported value in Table 3 is the average δ found by the SAT-based solver divided

Table 3: The ratio of δ found by the SAT-based solver and by the CBS-TU algorithm.

agents	U=1	U=3	U=5
2	1,00	0,50	0,56
4	0,77	0,40	0,46
6	0,76	0,46	0,40
8	0,79	0,39	0,36
10	0,75	0,42	0,41
12	0,75	0,34	0,40
14	0,80	0,39	0,36
16	0,71	-	0,33
18	0,72	-	0,57
20	0,71	-	-

by the average δ found by the CBS-TU algorithm. Therefore, the lower the number, the better solution the SAT-based solver produced. Notice that there are no settings with a value higher than 1 which is in compliance with Proposition 2.

6 CONCLUSION

In this paper, we presented a novel extension to the MAPF-TU problem by introducing a policy-based solution. Our approach addresses the limitations of plans handling uncertainties by leveraging an SAT-based model for policy generation, offering a robust and flexible alternative to traditional methods. We showed both theoretically and empirically that policies produce solutions with better quality, as measured by the length of each agent's path. We were also able to solve more instances within the given time limit than with the original search-based approach. Future works could explore hybrid approaches that combine policies with heuristics to improve computational efficiency.

ACKNOWLEDGMENTS

REFERENCES

Atzmon, D., Stern, R., Felner, A., Wagner, G., Barták, R., and Zhou, N. (2018). Robust multi-agent path finding. In *Proceedings of the Eleventh International Symposium on Combinatorial Search, SOCS*, pages 2–9. AAAI Press.

Barták, R. and Svancara, J. (2019). On sat-based approaches for multi-agent path finding with the sum-of-costs objective. In *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS*, pages 10–17. AAAI Press.

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., Oztok, U., and Schüller, P. (2013). A general formal framework for pathfinding problems with multiple agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 290–296.

Hönig, W., Kiesel, S., Tinka, A., Durham, J. W., and Ayanian, N. (2019). Persistent and robust execution of mapf schedules in warehouses. *IEEE Robotics and Automation Letters*, 4:1125–1131.

Hönig, W., Kumar, T., Cohen, L., Ma, H., Xu, H., Ayanian, N., and Koenig, S. (2016). Multi-agent path finding with kinematic constraints. In *Proceedings of the International Conference on Automated Planning and Scheduling, ICAPS*, volume 26, pages 477–485.

Morris, R., Chang, M. L., Archer, R., Cross, E. V., Thompson, S., Franke, J., Garrett, R., Malik, W., McGuire, K., and Hemann, G. (2015). Self-driving aircraft towing vehicles: A preliminary report. In *Workshops at the twenty-ninth AAAI conference on artificial intelligence*.

Ryan, M. (2010). Constraint-based multi-robot path planning. In *2010 IEEE International Conference on Robotics and Automation*, pages 922–928. IEEE.

Shahar, T., Shekhar, S., Atzmon, D., Saffidine, A., Juba, B., and Stern, R. (2021). Safe multi-agent pathfinding with time uncertainty. *Journal of Artificial Intelligence Research*, 70:923–954.

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

Shofer, B., Shani, G., and Stern, R. (2023). Multi agent path finding under obstacle uncertainty. In *Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling, ICAPS*, pages 402–410. AAAI Press.

Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T., Barták, R., and Boyarski, E. (2019). Multi-agent pathfinding: Definitions, variants, and benchmarks. In

Proceedings of the Twelfth International Symposium on Combinatorial Search (SOCS'19), pages 151–159. AAAI Press.

Surynek, P. (2010). An optimization variant of multi-robot path planning is intractable. In *Proceedings of the AAAI conference on artificial intelligence*, volume 24, pages 1261–1263.

Surynek, P. (2012). Towards optimal cooperative path planning in hard setups through satisfiability solving. In *Pacific Rim international conference on artificial intelligence*, pages 564–576. Springer.

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 *Proceedings of the Twenty-second European Conference on Artificial Intelligence (ECAI'16)*, pages 810–818. IOS Press.

Svancara, J., Atzmon, D., Strauch, K., Kaminski, R., and Schaub, T. (2024). Which objective function is solved faster in multi-agent pathfinding? it depends. In *Proceedings of the 16th International Conference on Agents and Artificial Intelligence, ICAART*, pages 23–33. SCITEPRESS.

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

Zhou, N. and Kjellerstrand, H. (2016). The picat-sat compiler. In *Proceedings of Practical Aspects of Declarative Languages - 18th International Symposium, PADL*, volume 9585 of *Lecture Notes in Computer Science*, pages 48–62. Springer.