Privacy Preserving Multi Agent Path Finding

Extended Abstract

Rotem Lev Lehman
Computer and Information Systems
Ben Gurion University of the Negev
Be'er Sheva, Israel
levlerot@post.bgu.ac.il

Roni Stern
Computer and Information Systems
Ben Gurion University of the Negev
Be'er Sheva, Israel
roni.stern@gmail.com

Guy Shani
MSAI, Microsoft
Ben Gurion University of the Negev
Be'er Sheva, Israel
guyshani@microsoft.com
shanigu@bgu.ac.il

ABSTRACT

In the multi-agent path finding (MAPF) problem, a group of agents search in a graph for a path for each agent where no two paths collide. This work considers MAPF applications in which the agents do not wish to share their paths due to privacy constraints. We formulate two types of privacy constraints in this context: *planning-level privacy* and *execution-level privacy*. The former means the agents cannot identify the planned location of the other agents, and the latter means agents cannot sense the location of each other during execution. We show a general approach to preserve planning-level privacy and show to how adapt two popular MAPF algorithms, namely PIBT and LaCAM, to preserve execution-level privacy. We also propose a post-processing technique that allows agents to reduce the costs of the returned solution without losing any privacy.

KEYWORDS

Multi-Agent Path Finding; Privacy Preserving Planning

ACM Reference Format:
Rotem Lev Lehman, Roni Stern, and Guy Shani. 2026. Privacy Preserving Multi Agent Path Finding: Extended Abstract. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 3 pages. https://doi.org/10.65109/JWHZ7620

1 MOTIVATION AND BACKGROUND

The Multi-Agent Path Finding (MAPF) problem arises when multiple mobile agents must each find a path from their respective start locations to their goal locations on a shared graph without colliding with each other. MAPF is motivated by a wide range of real-world applications, such as automated warehouse robotics, airport ground traffic management, and digital entertainment. In this work, we consider a *privacy-preserving* type of MAPF problems, where the agents must collaborate to avoid collisions, optimize a shared objective function, and still wish to keep some information private from each other. Use-cases for privacy-preserving MAPF include coordinated trucks and drones of different logistics companies, coordinating routes of human taxi drivers, and coordinating the movement of robots in shared environments.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/JWHZ7620

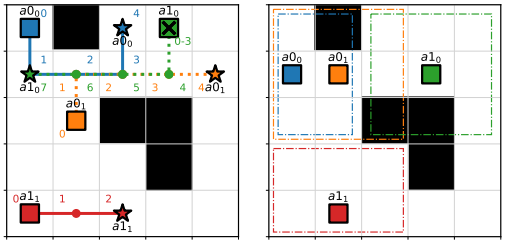

Figure 1: (left) A κPP solution for two agents and $k = 2$. (right) FoV of each agent at time step 1.

This form of privacy-preserving collaborative multi-agent planning has been studied in the context of other types of multi-agent planning problems, including Multi-Agent STRIPS [1, 8–10], Distributed Constraints Optimization (DCOP) [3, 5] and Distributed Constraint Satisfaction Problem (DisCSP) [20, 21]. Yet very limited work has been done on privacy-preserving MAPF [7]. Our setting somewhat similar to prior work on distributed methods for solving MAPF [6, 7, 14, 18, 19]. One such approach [2, 7] has each agent plan independently, coordinating and replanning online with other agents that enter its field of view. In that scheme, and in general in existing decentralized MAPF algorithms, agents may know the partial path of other agents if they are close to each other during the runtime. Thus, they do not provide any privacy guarantees, which is our primary objective.

Background. A classical MAPF problem with N agents is defined by a tuple $\langle G, s, t \rangle$ where $G = (V, E)$ is an undirected graph whose vertices are the possible locations agents may occupy, and every edge $(v, u) \in E$ represents that an agent can move from v to u without passing through any other vertex. The functions s and t map each agent to its initial and desired destination locations, respectively. Time is discretized into time steps. In every time step, each agent can either *wait* in its current location or *move* to a location adjacent to its current location. A *single-agent plan* for an agent i is a sequence of actions (wait or move) that if performed starting from $s(i)$ will end up in $t(i)$. A *joint plan* is a set of single-agent plans, one for each agent. For a joint plan Π, we denote by Π_i its constituent single-agent plan for agent i, and denote by $\Pi_i(t)$ as the vertex agent i is planned to occupy at timestep t according to Π_i. A pair of agents i and j have a *vertex conflict* in a joint plan Π if, according to their respective single-agent plans Π_i and Π_j, both agents are planned to occupy the same vertex at the same time. Similarly, agents have a *swapping conflict* in a joint plan if they are planned to swap locations over the same edge at the same time. A *valid* solution to a MAPF problem is a joint plan without

any conflict. The sum of costs (SOC) of a solution is the sum over the lengths of the single-agent plans in the solution.

Many algorithms have been proposed for solving classical MAPF [4, 17]. Prominent examples include CBS [15], which is optimal and solution-complete; LaCAM [11], which is complete; LaCAM* [12], which finds an initial solution and then continuously improve it as long as additional runtime is available; and PIBT [13], which is neither complete nor optimal yet is known to be very fast.

2 PROBLEM SETUP

In this work, we consider a group of agents faced with a classical MAPF problem $\langle G, s, t\rangle$. Each agent is fully aware of the underlying graph G, yet it does not know the initial and target location of the other agents. To coordinate, the agents may send messages to each other directly and immediately. However, each agent does not wish the agents to know where it plans to visit in every timestep on its way to its target. That is, for a MAPF solution Π, agent i, and time step t, the *private information* agent i does not wish to disclose is the location $\Pi_i(t)$. We consider two stages where such information can be revealed or inferred — during planning and during execution.

3 PLANNING-LEVEL PRIVACY

We denote by μ the set of messages passed between the agents during the planning process. The *agents' belief* about agent i at time t, denote $b_i(\mu, t)$, is the set of locations agent i may occupy at time t according to μ. In other words, $b_i(\mu, t)$ captures the uncertainty of the other agents regarding the true location of i at timestamp t. The *belief state* of the agents is the vector $b = (b_1, \ldots, b_N)$ containing all the agents' beliefs.

Definition 3.1 (k-Privacy). A belief state b is said to preserve *k-Privacy* if for every agent i and time step t, it holds that $|b_i^t(\mu)| \geq k$.

Intuitively, this means that for every agent i and time step t, the messages sent during planning are not sufficient to allow the other agents to narrow down the possible location of agent i to fewer than k possible locations.

Next, we propose the k-Privacy Preserving MAPF Planner (κPP), which is a general algorithm for solving a MAPF problem while ensuring the agents' belief state preserving k-privacy throughout planning. κPP works by having each agent i create a set of $k-1$ *mock agents*, each associated with a unique pair of initial and target locations. Then, the agents collaboratively solve a larger MAPF problem that includes non-conflicting single-agent plans for both real and mock agents. Finally, each (real) agent follows the single-agent plan created for it, discarding the plans created for the mock agents. Figure 1(left) illustrates a MAPF problem with two agents and the solution created for it by κPP for $k = 2$. The mock agents are marked in green and orange and the dashed lines mark the plans created for them.

A challenge in κPP is how each agent chooses the initial and target locations of its $k-1$ mock agents. One way to do this is to have each agent choose its group of locations randomly. A limitation for this approach is that two agents may collide in their assignments. Another approach is to model the problem of choosing the mock agents in a privacy-preserving manner as a Distributed CSP, for which there exists privacy-preserving solution methods [21]. Yet another approach for choosing mock agents is to use an *external dispatcher* agent, which dispatches mock agents that do not collide to all agents. Using such an external dispatcher can be viewed as some loss of privacy, but only if the dispatcher collaborates with one of the agents.

4 EXECUTION-LEVEL PRIVACY

Even if they execute a MAPF solution generated with κPP, the agent may still compromise their privacy during execution if the agents are equipped with sensors that allow them to sense the location of nearby agents. We formalize the sensing capability of an agent i by a *Field of View* (FoV) function F_i that maps every possible location v of agent i to the locations agent i can sense when situated in v.

Definition 4.1 (FoV Conflict). A pair of single agent plans Π_i and Π_j assigned to agents i and j, respectively, are said to have a FoV conflict if there exists t such that either $\Pi_i(t) \in FoV_j(\Pi_j(t))$ or $\Pi_j(t) \in FoV_i(\Pi_i(t))$.

A MAPF solution is said to preserve *Runtime k-Privacy* if the belief state generated from finding Π preserves k-Privacy and there are no *FoV* conflicts in Π. To obtain a Runtime k-Privacy solution, we propose to use κPP and modify the underlying MAPF planner to incorporate additional *FoV* conflicts. These *FoV* conflicts are enforced only between different agent groups (where an agent group is a real agent and its mock agents), since visibility among agents of the agent group does not compromise privacy during execution. Figure 1(right) shows the *FoV* of the agents at time step 1. Notice that the *FoV* of agents a_{0_0} and a_{0_1} contain the other agent, but it is fine since they are both in the same agent-group.

Improving Solution Quality. In a runtime k-privacy solution, all the locations planned for agent i and associated $k-1$ mock agents will not be in the FoV of any other agent at the same time. We refer to such vertices, i.e., the vertices that are guaranteed to not be sensed by any other agents, as the *safe zone* of agent i at timestep t. Each agent can choose a different plan for itself without coordinating with the other agents as long as the new plan only occupies vertices in corresponding safe zones. This can be beneficial, as the agent may now prioritize its single-agent plan over those of its $k-1$ mock agents, potentially obtaining lower SOC for itself, without compromising privacy. Moreover, the agents can split between themselves the locations not in any safe zone cells, obtaining an extended safe zone for each agent.

5 CONCLUSION AND FUTURE WORK

We introduce a new aspect to MAPF research: planning-level privacy and execution-level privacy. We formalized these notions and proposed methods for achieving them, based on existing MAPF solvers. We also proposed a post-processing mechanism for improving the quality of the solutions by defining for each agent *safe-zones* where it can freely re-plan. We implemented privacy-preserving solvers as outlines in this work based on PIBT and LaCAM*. An evaluation on standard MAPF benchmarks [16] for different values of k and sensing ranges showed that, as expected, increasing k and sensing ranges drastically increase runtime and solution quality in most cases. Future work will extend this experimental evaluation, develop efficient suboptimal and optimal privacy-preserving MAPF solvers, and explore tradeoffs between privacy, runtime, and cost.

ACKNOWLEDGMENTS

This research was funded by ISF grant No. 1238/23 to Roni Stern, and by the ministry of science grant No. 0008612/1001948158.

REFERENCES

[1] Ronen I Brafman. 2015. A privacy preserving algorithm for multi-agent planning and search. In *24th International Joint Conference on Artificial Intelligence, IJCAI 2015*. International Joint Conferences on Artificial Intelligence, 1530–1536.

[2] Stepan Dergachev and Konstantin Yakovlev. 2021. Distributed multi-agent navigation based on reciprocal collision avoidance and locally confined multi-agent path finding. In *IEEE International Conference on Automation Science and Engineering (CASE)*. 1489–1494.

[3] Boi Faltings, Thomas Léauté, and Adrian Petcu. 2008. Privacy guarantees through distributed constraint satisfaction. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, Vol. 2. IEEE, 350–358.

[4] Ariel Felner, Roni Stern, Solomon Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek. 2017. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In *International Symposium on Combinatorial Search*, Vol. 8. 29–37.

[5] Rachel Greenstadt, Barbara Grosz, and Michael D Smith. 2007. SSDPOP: improving the privacy of DCOP with secret sharing. In *International joint conference on Autonomous agents and multiagent systems (AAMAS)*. 1–3.

[6] Florence Ho, Rúben Geraldes, Artur Gonçalves, Bastien Rigault, Benjamin Sportich, Daisuke Kubo, Marc Cavazza, and Helmut Prendinger. 2020. Decentralized multi-agent path finding for UAV traffic management. *IEEE Transactions on Intelligent Transportation Systems* 23, 2 (2020), 997–1008.

[7] M Onur Keskin, Furkan Cantürk, Cihan Eran, and Reyhan Aydoğan. 2024. Decentralized multi-agent path finding framework and strategies based on automated negotiation. *Autonomous Agents and Multi-Agent Systems* 38, 1 (2024), 10.

[8] Rotem Lev Lehman, Guy Shani, and Roni Stern. 2022. Reducing disclosed dependencies in privacy preserving planning. *Autonomous Agents and Multi-Agent Systems* 36, 2 (2022), 52.

[9] Shlomi Maliah, Guy Shani, and Roni Stern. 2017. Collaborative privacy preserving multi-agent planning: Planners and heuristics. *Autonomous agents and multi-agent systems* 31, 3 (2017), 493–530.

[10] Raz Nissim and Ronen Brafman. 2014. Distributed heuristic forward search for multi-agent planning. *Journal of Artificial Intelligence Research* 51 (2014), 293–332.

[11] Keisuke Okumura. 2023. LaCAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*.

[12] Keisuke Okumura. 2024. Engineering LaCAM*: Towards Real-Time, Large-Scale, and Near-Optimal Multi-Agent Pathfinding. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

[13] Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. 2022. Priority Inheritance with Backtracking for Iterative Multi-agent Path Finding. *Artificial Intelligence* (2022), 103752. https://doi.org/10.1016/j.artint.2022.103752

[14] Poom Pianpak, Tran Cao Son, Phoebe O Toups Dugas, and William Yeoh. 2019. A distributed solver for multi-agent path finding problems. In *International Conference on Distributed Artificial Intelligence (DAI)*. 1–7.

[15] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. 2015. Conflict-based search for optimal multi-agent pathfinding. *Artificial intelligence* 219 (2015), 40–66.

[16] Roni Stern, Nathan Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Kumar, et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*. 151–158.

[17] Pavel Surynek. 2022. Problem Compilation for Multi-Agent Path Finding: a Survey.. In *IJCAI*. 5615–5622.

[18] Prasanna Velagapudi, Katia Sycara, and Paul Scerri. 2010. Decentralized prioritized planning in large multirobot teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. 4603–4609.

[19] Hanlin Wang and Michael Rubenstein. 2020. Walk, stop, count, and swap: decentralized multi-agent path finding with theoretical guarantees. *IEEE Robotics and Automation Letters* 5, 2 (2020), 1119–1126.

[20] M. Yokoo, E.H. Durfee, T. Ishida, and K. Kuwabara. 1998. The distributed constraint satisfaction problem: formalization and algorithms. *IEEE Transactions on Knowledge and Data Engineering* 10, 5 (1998), 673–685. https://doi.org/10.1109/69.729707

[21] Makoto Yokoo, Koutarou Suzuki, and Katsutoshi Hirayama. 2002. Secure distributed constraint satisfaction: Reaching agreement without revealing private information. In *International Conference on Principles and Practice of Constraint Programming*. Springer, 387–401.