Enhancing Lifelong Multi-Agent Path-finding by Using Artificial Potential Fields

Extended Abstract

Arseniy Pertzovsky
Ben-Gurion University of the Negev
Beer Sheva, Israel
arsenip@post.bgu.ac.il

Roni Stern
Ben-Gurion University of the Negev
Beer Sheva, Israel
roni.stern@gmail.com

Ariel Felner
Ben-Gurion University of the Negev
Beer Sheva, Israel
felner@bgu.ac.il

Roie Zivan
Ben-Gurion University of the Negev
Beer Sheva, Israel
zivanr@bgu.ac.il

ABSTRACT

We explore the use of Artificial Potential Fields (APFs) to solve Lifelong Multi-Agent Path Finding (LMAPF) problems. In LMAPF, a team of agents must move to their goal locations without collisions, and new goals are generated upon arrival. We propose methods for incorporating APFs in a range of LMAPF algorithms, including Prioritized Planning and MAPF-LNS2. Experimental results show that using APF yields up to a 7-fold increase in overall system throughput for LMAPF.

KEYWORDS

Multi-agent Pathfinding, Artificial Potential Fields, Multi-robot Path Planning

ACM Reference Format:
Arseniy Pertzovsky, Roni Stern, Ariel Felner, and Roie Zivan. 2025. Enhancing Lifelong Multi-Agent Path-finding by Using Artificial Potential Fields: Extended Abstract. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 3 pages.

1 INTRODUCTION

Artificial Potential Fields (APFs) [4] is a physics-inspired approach for the deployment of mobile agents in an environment with obstacles. Typically, obstacles and other agents exert repulsive forces while the goal applies an attractive force [5]. APFs have been used to solve many motion planning problems [2, 3, 10], with successful applications in obstacle avoidance of an unmanned aircraft [16], collision avoidance systems for automated vehicles [21] and more [10]. In this work, we explore how APFs can be used to solve Multi-Agent Pathfinding (MAPF) problems.

MAPF and *lifelong* MAPF (LMAPF) are the problems of finding a set of paths for a group of agents such that if each agent follows its path, it ends up in its goal location without colliding with any other agent [19]. LMAPF is an online MAPF [20] variant in which an agent receives a new goal whenever it reaches its current goal location. Instances of MAPF exist in many fields [1, 9, 12, 17, 22]. When solving LMAPF, congested areas tend to become more and more congested over time, resulting in a decrease in overall system efficiency. Using APFs is a natural approach to encourage agents to avoid such areas, adding repulsion forces not only to avoid obstacles but also to avoid the paths of other agents. We explored the use of APFs within existing MAPF solvers. We consider using single-agent pathfinding algorithms to plan for individual agents under different types of constraints. Temporal A* (TA*) [18] and SIPPS [7] are the most prevalent examples of such algorithms. We propose modified versions of these algorithms, called TA*+APF and SIPPS+APF, that change the cost of agents' actions to take into consideration the APFs created by other agents who have already planned their paths.

We evaluated our approaches experimentally on a range of standard MAPF benchmark problems. The results show that our APF-augmented algorithms are very effective when solving LMAPF, yielding up to a 7-fold increase in overall system throughput.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

2 DEFINITION AND BACKGROUND

A classical MAPF problem is defined by a tuple $\langle k, G, s, g \rangle$ where k is the number of agents, $G = (V, E)$ represents an undirected graph, s and g map an agent to its start and goal vertices. In every time-step, each agent occupies a single vertex and performs a single *action*, which is a function $a : V \rightarrow V$ such that $a(v) = v'$. A *single-agent path* for agent a_i, denoted π_i, is a sequence of actions π_i that is applicable starting from s_i and ends up in g_i. A solution to a MAPF is a set of single-agent paths $\pi = \{\pi_1, \ldots, \pi_k\}$, one per agent, that do not *conflict*, meaning no two agents switch their locations or occupy the same vertex at the same time. *Lifelong MAPF (LMAPF)* [6] is an important type of *online MAPF* [20], where agents continuously receive new tasks from a task assigned. The efficiency of LMAPF algorithms is usually measured by the overall system *throughput* achieved when using them, which is measured by the number of tasks fulfilled in a given period of time [8, 11].

3 TEMPORAL A* WITH APFS

We propose to use APFs in TA* such that the resulting path not only avoids collisions with the paths of other agents (which is a hard constraint) but also attempts to keep distance from them by

considering the repulsion of their APFs. We refer to our TA* variant as *Temporal A* with APFs* (TA*+APF).

To bias the resulting path to keep distance from these paths, TA*+APF creates for every path $\pi_i \in \{\pi_1, \ldots, \pi_{k'}\}$ a repulsion APF function APF_i that maps every location-time pair (v, t) to a real number representing the added penalty incurred by planning to occupy v at time t. The APF induced by agent a_i on location v and time t is computed as follows:

$$APF_i(v,t) = \begin{cases} 0 & \text{if } d(v, \pi_i[t]) \geq d_{max} \\ w \cdot \gamma^{-d(v,\pi_i[t])} & \text{otherwise} \end{cases} \quad (1)$$

where d_{max}, γ, and w are predefined parameters and $d(v, t, \pi_i[t])$ is the minimal distance between v and $\pi_i[t]$. The w parameter controls the strength of the repulsion, γ controls the rate of decay, i.e., how fast its intensity of the repulsion declines while moving away from its source, and d_{max} defines how far away from v_i the repulsion affects the cost. To aggregate all the APFs we used a simple sum. That is, the APF cost of moving into location v at time is

$$cost_{APF}(v,t) = \sum_{i \in \{1,\ldots,k'\}} APF_i(v,t) \quad (2)$$

We chose to use our APF-inspired cost function when computing the g value of a search node, as follows. Let $cost(parent, n)$ be the cost of moving the agent from *parent* to n, and let (v, t) be the vertex and time-step that node n represents. In TA*+APF we compute the $g(n)$ as follows:

$$g(n) = g(parent) + cost(parent, n) + cost_{APF}(v,t) \quad (3)$$

TA*+APF runs TA* according to $f(n) = g(n) + h(n)$, using this modified $g(n)$ function.

4 SIPPS WITH APFS

A more recent state-of-the-art low-level solver in multiple MAPF algorithms is SIPPS [7]. First, SIPPS sorts the open list according to $c(n)$, which tracks the number of soft collisions (violated soft constraints caused by a collision with another path). Then, the secondary sort (between the nodes with the same $c(n)$ value) is executed according to $f(n)$ as in TA*. The $g(n)$ component equals the lowest value in the node's time interval. This way, SIPPS ensures that the found path minimizes the number of soft collisions. We refer to our SIPPS variant as *SIPPS with APFs* (SIPPS+APF). Because the sorting of the open list in SIPPS+APF is executed according to two components, first $c(n)$ and then $f(n)$, we explored incorporating APFs separately in each of these components. The final formal definition is presented as follows.

$$cost_{APF}(n) = \max_{t \in [t^n_{start}, t^n_{end})} \sum_{i \in \{1,\ldots,k'\}} APF_i(v,t) \quad (4)$$

Where $APF_i(v, t)$ is defined in TA*+APF, and $[t^n_{start}, t^n_{end})$ is the current safe interval of the n node. t^n_{start} and t^n_{end} are the beginning and the end time-steps of the interval. In other words, $cost_{APF}(n)$ represents the highest APFs an agent can encounter during its time interval. In SIPPS+APF we compute the $g(n)$ and $c(n)$ as follows:

$$g(n) = t^n_{start} + cost_{APF}(n) \quad (5)$$

$$c(n) = count\_soft\_collisions(n) + cost_{APF}(n) \quad (6)$$

SIPPS+APF runs SIPPS and first prioritizes nodes according to small $c(n)$ values. In case of a tie, it moves to the secondary priority and prefers nodes with smaller $f(n) = g(n) + h(n)$.

5 EXPERIMENTAL STUDY

We conducted an experimental evaluation comparing the use of APFs within PrP [18], LNS2 [7], PIBT [15], LaCAM [13], and LaCAM* [14], where PrP and LNS2 are implemented twice: once with TA* and once with SIPPS. All experiments were performed on four different maps from the MAPF benchmark [19]. The number of agents used in our experiments varied from 50 to 450. We executed 25 random instances per every number of agents, map, and algorithm. The APF parameters used were $w = 1$, $d_{max} = 4$, and $\gamma = 2$ for TA*+APF, and $w = 0.1$, $d_{max} = 3$, and $\gamma = 3$ for SIPPS+APF. [1]

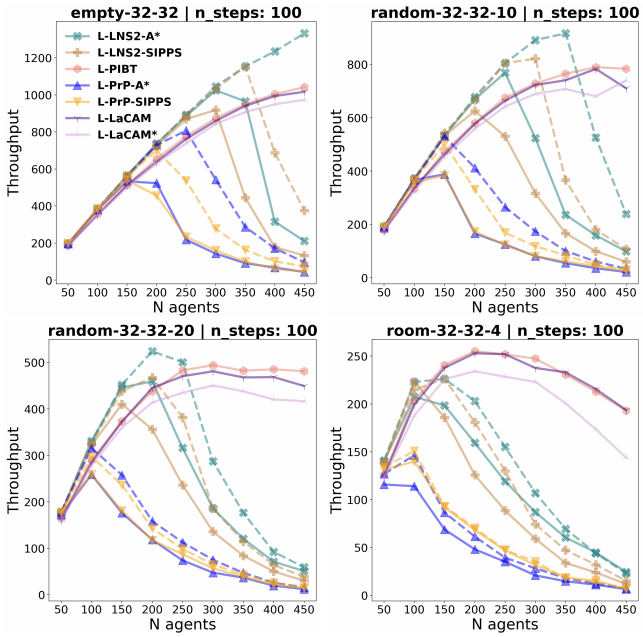

Figure 1: LMAPF: Average Throughput. Dashed lines - APF-enhanced; Solid lines - no APFs

As can be seen in Figure 1, the use of APFs significantly increases the throughput of all other algorithms in all maps. For example, in the *empty-32-32* grid, LNS2 with TA*+APF reaches a throughput of approximately 1400 with 450 agents, which is approximately 7 times more than the throughput of vanilla LNS2 for the same number of agents, and it significantly outperforms other baselines.

6 CONCLUSION AND FUTURE WORK

We investigated whether MAPF can be solved efficiently using artificial potential fields (APFs). We proposed using APFs in TA* and in SIPPS, key components of many MAPF algorithms. In the future, we want to explore more efficient ways to incorporate APFs into PIBT and LaCAM algorithms.

[1] The adaptation of APFs for PIBT, LaCAM, and LaCAM* did not result in any good performance. Hence we do not report it here.

REFERENCES

[1] Roman Barták, Jiří Švancara, Věra Škopková, David Nohejl, and Ivan Krasičenko. 2019. Multi-agent path finding on real robots. *AI Communications* (2019).

[2] Robert Daily and David M Bevly. 2008. Harmonic potential field path planning for high speed vehicles. In *2008 American Control Conference*. IEEE, 4609–4614.

[3] Johan Hagelback and Stefan Johansson. 2009. A multi-agent potential field-based bot for a full RTS game scenario. In *AAAI*, Vol. 5. 28–33.

[4] Oussama Khatib. 1986. The potential field approach and operational space formulation in robot control. In *Adaptive and Learning Systems: Theory and Applications*. Springer, 367–377.

[5] Yoram Koren and Johann Borenstein. 1991. Potential field methods and their inherent limitations for mobile robot navigation. In *ICRA*. 1398–1404.

[6] Jiaoyang Li, Zhe Chen, Daniel Harabor, P Stuckey, and Sven Koenig. 2021. Anytime multi-agent path finding via large neighborhood search. In *IJCAI*.

[7] Jiaoyang Li, Zhe Chen, Daniel Harabor, Peter J Stuckey, and Sven Koenig. 2022. MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search. In *AAAI*.

[8] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W. Durham, T. K. Satish Kumar, and Sven Koenig. 2021. Lifelong Multi-Agent Path Finding in Large-Scale Warehouses. *AAAI* (May 2021).

[9] Hang Ma, Jingxing Yang, Liron Cohen, T. K. Satish Kumar, and Sven Koenig. 2017. Feasibility Study: Moving Non-Homogeneous Teams in Congested Video Game Environments. In *AIIDE*.

[10] Thi Thoa Mac, Cosmin Copot, Duc Trung Tran, and Robin De Keyser. 2016. Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems* 86 (2016), 13–28.

[11] Jonathan Morag, Roni Stern, and Ariel Felner. 2023. Adapting to Planning Failures in Lifelong Multi-Agent Path Finding. In *SoCS*.

[12] Robert Morris, Corina S Pasareanu, Kasper Søe Luckow, Waqar Malik, Hang Ma, TK Satish Kumar, and Sven Koenig. 2016. Planning, Scheduling and Monitoring for Airport Surface Operations.. In *AAAI Workshop: Planning for Hybrid Systems*.

[13] Keisuke Okumura. 2023. Lacam: Search-based algorithm for quick multi-agent pathfinding. In *AAAI*, Vol. 37. 11655–11662.

[14] Keisuke Okumura. 2023. LaCAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding. *AAAI* (Jun. 2023).

[15] Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. 2022. Priority inheritance with backtracking for iterative multi-agent path finding. *AI* 310 (2022), 103752.

[16] Hamed Rezaee and Farzaneh Abdollahi. 2012. Adaptive artificial potential field approach for obstacle avoidance of unmanned aircrafts. In *AIM*. IEEE, 1–6.

[17] Oren Salzman and Ron Zvi Stern. 2020. Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems blue sky ideas track. In *AAMAS*.

[18] David Silver. 2005. Cooperative Pathfinding. In *AIIDE*.

[19] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Bartak. 2019. Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks. In *SoCS*. 151–158.

[20] Jiří Švancara, Marek Vlk, Roni Stern, Dor Atzmon, and Roman Barták. 2019. Online multi-agent pathfinding. In *AAAI*.

[21] Nurbaiti Wahid, Hairi Zamzuri, Mohd Azizi Abdul Rahman, Satoshi Kuroda, and Pongsathom Raksincharoensak. 2017. Study on potential field based motion planning and control for automated vehicle collision avoidance systems. In *ICM*. 208–213.

[22] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. 2008. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine* 29, 1 (2008), 9–9.