Scalable Robust Multi-Agent Path Finding (Student Abstract)

Amit Bouzaglo

Bar-Ilan University
bouzaga1@biu.ac.il

Introduction and Background

Multi-Agent Path Finding (MAPF) (Stern et al. 2019) receives as input an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and lists of start and goal vertices for n agents $S = (s_1, \ldots, s_n)$ and $G = (g_1, \ldots, g_n)$, respectively. *Plan* $\Pi = (\pi_1, \ldots, \pi_n)$ is a list of paths, where each *path* $\pi_i \in \Pi$ starts with s_i and ends with g_i, composed of *move* and *wait* actions. Two paths *conflict* when agents simultaneously occupy a vertex or traverse an edge in opposite directions. A *solution* is a conflict-free plan; cost $c(\pi_i) = |\pi_i| - 1$ and $C(\Pi) = \sum_i c(\pi_i)$. While MAPF is NP-hard (Yu and LaValle 2013; Surynek 2010), advanced algorithms can solve it for many agents (Sharon et al. 2015; Maliah, Atzmon, and Felner 2025).

MAPF algorithms operate in either a *vertex configuration space* (VCS), where each node m represents agents' current vertices. For m, we estimate the cost of reaching a goal for each agent separately, ignoring all other agents; $h(m)$ is the sum of all these estimates. In a *path configuration space* (PCS), each node represents a set of (possibly conflicting) paths.

LaCAM (Okumura 2023b) quickly finds suboptimal MAPF solutions via two search levels. The high level performs a depth-first search in VCS, starting from a *Root* node containing the start vertices of the agents (start configuration), iteratively and partially expanding nodes—only one successor is generated per expansion, and a node is removed from the stack only when all its successors have been generated. The low level uses a constraint tree (root contains an empty constraint set; each expansion chooses an unconstrained agent and creates a child node per possible action with a positive constraint) to enumerate configurations for each successor, calling *PIBT* (Okumura et al. 2022) as a configuration generator that efficiently moves agents toward their goals. LaCAM halts when the first goal configuration is found.

*LaCAM** (Okumura 2023a) extends LaCAM into an anytime solver converging to the optimal solution. It searches VCS via depth-first branch-and-bound, pruning nodes m with $f(m) = g(m) + h(m) \geq U$ (U = incumbent cost). LaCAM* halts only after scanning the entire search tree, thus guaranteeing solution optimality. An improved version (Okumura 2024) adds *Space utilization optimization* (SUO/Scatter), which prefers configurations that scatter agents to prevent congestion in narrow spaces.

Conflict-Based Search (CBS) (Sharon et al. 2015), a common optimal MAPF algorithm, searches in PCS (unlike LaCAM and LaCAM*), planning paths and iteratively resolving conflicts until a solution is found.

While MAPF solutions are conflict-free, conflicts may arise during execution due to unexpected delays. *k-Robust MAPF* (kRMAPF) (Atzmon et al. 2020) extends MAPF to allow each agent to delay up to k times. When a delay occurs, the agent remains in place instead of performing a move action. A *k-delay conflict* between paths π_i and π_j occurs when $\pi_i(t_1) = \pi_j(t_2)$ with $|t_1 - t_2| \leq k$. A kRMAPF solution contains no k-delay conflicts.

To optimally solve kRMAPF, *kRCBS* (Atzmon et al. 2020) was proposed and later improved via pairwise symmetry-breaking constraints, which efficiently find compatible and optimal paths for pairs of conflicting agents (Chen et al. 2021). However, even with this improvement, kRCBS does not scale to many agents.

In this paper, we adjust LaCAM* to solve kRMAPF (kRLaCAM*) and propose a *cardinality reduction* (CR) improvement (kRLaCAM*-CR), which reduces the possible actions available to the low-level solver and configuration generator. We experimentally show that kRLaCAM*-CR significantly outperforms kRCBS and scales to a larger number of agents without a considerable increase in solution cost.

k-Robust LaCAM* (kRLaCAM*)

There are two main modifications required to adjust LaCAM* to find k-robust solutions.

The first is its state space. kRLaCAM* uses an extended VCS where each node m represents agents' current positions as well as the k previous configurations occupied along their paths. Maintaining the k previous configurations is important for optimality, and a node is pruned if it contains conflicting configurations. This modification was also proposed by Atzmon et al. (2020).

The second modification allows all agents to simultaneously wait at the low level. Consider the example in Figure 1 with three agents and $k = 2$. The optimal solution $\Pi = (\pi_1, \pi_2, \pi_3)$ is $\pi_1 = (s_1, A, A, s_2, g_1)$, $\pi_2 =$

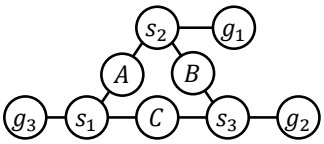

Figure 1: Example where all agents simultaneously wait.

(s_2, B, B, s_3, g_2), and $\pi_3 = (s_3, C, C, s_1, g_3)$ with $C(\Pi) = 12$, in which all agents perform a wait action at timestep 1. This case is not naturally allowed in LaCAM* but is required in kRLaCAM* to guarantee convergence to the optimal solution.

Cardinality Reduction Improvement

When a high-level node is (partially) expanded by kRLaCAM*, a new configuration is needed for its new high-level successor. For this successor, the low level may generate a configuration containing k-delay conflicts, causing the successor to be immediately pruned, a new low-level constraint set, and a new configuration generated, until a conflict-free configuration is created. To prevent the low level from generating such conflicting configurations, for each agent we detect actions that conflict with the previous k configurations and remove them from the possible actions provided to both the low-level search and the configuration generator. Therefore, new high-level nodes may only contain regular (0-delay) conflicts but no k-delay ($k > 0$) conflicts. We denote this improvement as *cardinality reduction* (kRLaCAM*-CR).

Experimental Study

We perform a preliminary experimental study evaluating kRCBS and kRLaCAM*-CR on the *brc202d* benchmark map from the *MovingAI* repository (Stern et al. 2019) with 5, 10, 15, 20, and 25 randomly allocated agents (25 problem instances each), a 90-second time limit, measuring success rate and average cost over instances solved by both solvers. Results are in Figures 2 and 3. No significant difference was observed across k values; each curve in Figure 3 represents a given agent count and k, with shading showing cost variation across instances.

kRCBS solves fewer instances as agents or k increase, whereas kRLaCAM*-CR solved all instances for any tested value. For both solvers, cost increases with agent count; however, as agents increase, kRLaCAM*-CR's cost grows farther from the optimal achieved by kRCBS.

Conclusion

In this paper, we adjust LaCAM* for finding robust MAPF solutions (kRLaCAM*) and improve it by reducing the cardinality of options available to its low-level solver and configuration generator (kRLaCAM*-CR). Empirically, kRLaCAM*-CR outperforms kRCBS in success rate, but achieves solutions with higher cost.

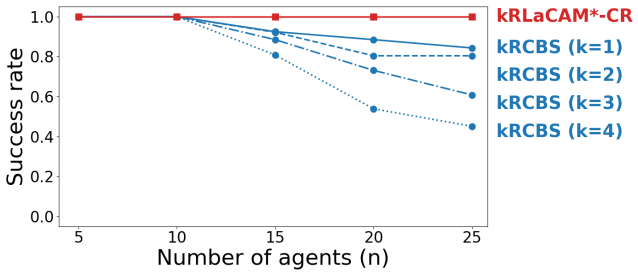

Figure 2: Success rate of kRCBS and kRLaCAM*-CR.

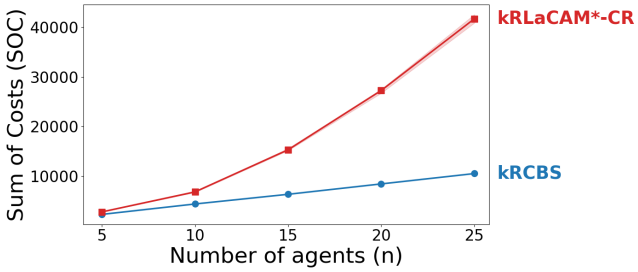

Figure 3: Average cost of kRCBS and kRLaCAM*-CR.

References

Atzmon, D.; Stern, R.; Felner, A.; Wagner, G.; Barták, R.; and Zhou, N.-F. 2020. Robust multi-agent path finding and executing. *JAIR*, 67: 549–579.

Chen, Z.; Harabor, D. D.; Li, J.; and Stuckey, P. J. 2021. Symmetry breaking for k-robust multi-agent path finding. In *the AAAI Conference on Artificial Intelligence (AAAI)*, 12267–12274.

Maliah, A.; Atzmon, D.; and Felner, A. 2025. Minimizing Makespan with Conflict-Based Search for Optimal Multi-Agent Path Finding. In *AAMAS*, 1418–1426.

Okumura, K. 2023a. Improving LaCAM for scalable eventually optimal multi-agent pathfinding. In *IJCAI*, 243–251.

Okumura, K. 2023b. LaCAM: search-based algorithm for quick multi-agent pathfinding. In *AAAI*, 11655–11662.

Okumura, K. 2024. Engineering LaCAM*: Towards Real-time, Large-scale, and Near-optimal Multi-agent Pathfinding. In *AAMAS*, 1501–1509.

Okumura, K.; Machida, M.; Défago, X.; and Tamura, Y. 2022. Priority inheritance with backtracking for iterative multi-agent path finding. *AIJ*, 310: 103752.

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

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

Surynek, P. 2010. An Optimization Variant of Multi-Robot Path Planning Is Intractable. In *AAAI*, 1261–1263.

Yu, J.; and LaValle, S. M. 2013. Structure and Intractability of Optimal Multi-Robot Path Planning on Graphs. In *AAAI*, 1444–149.