LaCAM* Variants for Minimizing Makespan in Multi-Agent Path Finding (Extended Abstract)

Omer Idgar[1], **Dor Atzmon**[2], **Ariel Felner**[1]
[1]Ben-Gurion University
[2]Bar-Ilan University
idgaro@post.bgu.ac.il, dor.atzmon@biu.ac.il, felner@bgu.ac.il

1 Introduction and Background

Multi-Agent Path Finding (MAPF) (Stern et al. 2019) is the problem of finding paths for multiple agents (a plan) from their start to goal vertices without conflicts. Two common cost functions for solutions are: (1) *sum-of-costs* (SOC), which is equal to the sum of the costs of the paths, and (2) *makespan* (MKS), which is equal to the longest path.

LaCAM (Okumura 2023b) is a MAPF algorithm, capable of quickly finding suboptimal solutions. On the high level, LaCAM is executed in a state space, where each node represents the vertices occupied by the agents, starting from a root node containing the start vertices of the agents (start configuration). The high level performs a *depth-first search* (using a LIFO stack), and iteratively takes the node from the head of the stack and partially expands it – only a single successor is generated, and the node remains in the stack. A node is removed from the LIFO stack when all its successors have been generated. The low level chooses a (conflict-free) configuration for the next successor, often using *PIBT* (Okumura et al. 2022), a fast MAPF algorithm that efficiently moves the agents towards their goals. PIBT gives higher priority to agents with longer paths. LaCAM halts when the first goal configuration is found. *LaCAM** (Okumura 2023a) is a recent anytime solver, which extends LaCAM and converges to the optimal MAPF solution. On the high level, LaCAM* searches in a *depth-first branch and bound* (DFBnB) manner (denoted LaCAM*-BnB) and halts only after it finishes scanning the search tree, thus guaranteeing optimality.

When LaCAM*'s low-level chooses a configuration, e.g., by PIBT, the agents may eventually become congested in narrow spaces. Therefore, recently, an improved version of LaCAM* was introduced (Okumura 2024) (denoted LaCAM*2), which has a few modifications. The main modification, called *Space utilization optimization* (SUO/Scatter), prevents multiple agents from reaching crowded areas by preferring configurations that better scatter the agents. While LaCAM*2 finds a (suboptimal) solution quickly, it converges to the optimal solution very slowly (for SOC).

In this paper, we show that, LaCAM* can quickly optimally solve MAPF for MKS. As we explain, unlike SOC, the heuristic estimate for MKS is very accurate, especially on large and sparse maps. We experimentally demonstrate that, in such sparse maps, LaCAM* significantly outperforms CBSM (Maliah, Atzmon, and Felner 2025), a CBS-based (Sharon et al. 2015) algorithm specifically designed for MKS, making LaCAM* the state-of-the-art optimal MAPF solver for MKS on these maps. We also consider LaCAM* that searches in a IDA* manner (denoted LaCAM*-IDA*). Experimentally, LaCAM* with IDA* often significantly outperforms LaCAM* with DFBnB and CBSM in finding optimal MKS solutions, presenting state-of-the-art performance on various benchmark maps.

2 Finding Optimal Solutions with LaCAM*

LaCAM searches in a DFS manner to find a (suboptimal) solution as fast as possible (the original paper was titled *"LaCAM: Search-Based Algorithm for Quick Multi-Agent Pathfinding"*) (Okumura 2023b). Following the same direction, LaCAM* (titled *"Improving LaCAM for Scalable Eventually Optimal Multi-Agent Pathfinding"*) (Okumura 2023a) searches in a DFBnB manner. It initially finds a solution and repeatedly improves the solution quality until the optimal solution is proven. LaCAM* aimed to be scalable (the author mentions that *"it [LaCAM*] suboptimally solved 99% of the instances retrieved from the MAPF benchmark"*) but it did not excel in converging to the optimal solution (*"the convergence speed was slow in large instances with many agents"*). All this was reported and tested for SOC. However, *what about MKS?* We experimentally examined the difference in finding optimal solutions for SOC and MKS with LaCAM*2-BnB on 50 problem instances of the *warehouse-20-40-10-2-1* map (Stern et al. 2019), with 100, 200, ..., 1,000 agents. As expected, LaCAM*2-BnB could not optimally solve any instance for SOC. However, LaCAM*2-BnB optimally solved many instances for MKS (even 32 instances with 1,000 agents). These findings inspire the use of LaCAM* as an optimal algorithm for MKS.

3 Experimental Study

We experimented (on an Intel® Core Ultra 9-185H CPU (8 cores) with 16GB RAM) with CBSM, LaCAM*-BnB, LaCAM*-IDA*, LaCAM*2-BnB, and LaCAM*2-IDA* on four representative benchmark maps from the *MovingAI* repository (Stern et al. 2019): *empty-32-32* (Empty),

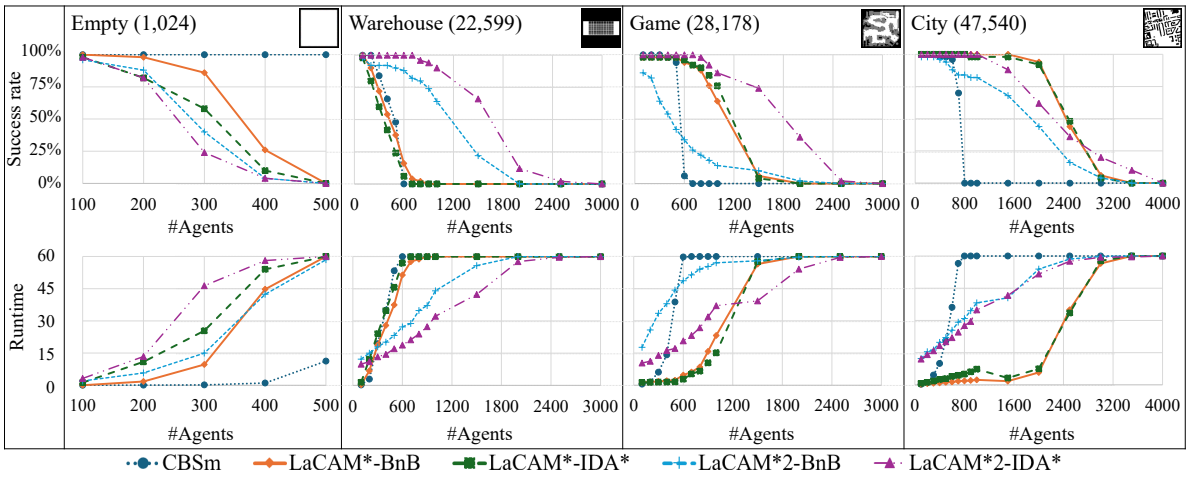

Figure 1: Success rate (top) and average runtime in seconds (bottom) on four benchmark maps. Time limit 60 seconds

warehouse-20-40-10-2-1 (Warehouse), *den520d* (Game), and *Berlin_1_256* (City). We generated problem instances with the *MAPF-LNS2* (Li et al. 2022) instance generator using the publicly available implementation. For each map, 50 instances were created with 100, . . . , 1,000, and then 1,500, . . . , 4,000 agents. Our implementation is available at https://github.com/OmerIdgar/LaCAM-Makespan-Versions. Figure 1 shows the success rate (top) and average runtime (bottom); the time limit per instance is 60 seconds.

CBSm vs. LaCAM*. CBSm outperformed the LaCAM* solvers in the small map (Empty). A similar trend was observed by Maliah, Atzmon, and Felner (2025). Also, the IDA* variants may need to perform multiple iterations. Thus, their DFBnB counterparts might perform better. By contrast, in the three larger maps, the LaCAM* versions outperformed CBSm, highlighting that, unlike previously thought, LaCAM* is a strong algorithm for optimally solving MAPF for MKS. Importantly, an IDA* variant of LaCAM* was almost always the best algorithm on the larger maps. In fact, LaCAM*2-IDA* optimally solved instances containing thousands of agents for the first time (e.g., in City with 3,500 agents). In general, in all LaCAM*'s optimally solved instances, the heuristic at the root was perfect.

DFBnB vs. IDA*. In the larger maps, LaCAM*2-IDA* outperformed LaCAM*2-BnB. Unlike LaCAM*, the low-level of LaCAM*2 chooses configurations that better scatter the agents to sparse areas. For LaCAM*2-BnB, when scattering the agents, the f-value of successors often increases. This results in a longer runtime for DFBnB before finding the optimal solution. However, LaCAM*2-IDA* only considers nodes with minimal f-values. Therefore, when a successor of a higher f-value is created, it forces the nodes to generate more successors until one with the minimal f-value is generated. As a result, LaCAM*2-IDA* scatters the agents but restrains this behavior by choosing a low-cost node.

Comparing IDA* variants. In the midsize maps (Warehouse, Game), LaCAM*2-IDA* was better than LaCAM*-IDA*, mainly because LaCAM*2's low level chooses configurations that scatter the agents to sparse areas, and a solution can be quickly found. On the larger (and sparser) map (City), there was no need for scattering and, in many cases, LaCAM*-IDA* outperformed LaCAM*2-IDA*.

To summarize: On small dense maps, h is inaccurate; CBSm should be used. On larger, sparser maps, h is often perfect; LaCAM*2-IDA* is recommended. But, for very sparse maps, LaCAM*-IDA* is also recommended.

Acknowledgments

This work was supported by ISF grants #909/23 and #1511/25. This work was also supported by MOST grant #6908 and by a MOST grant awarded to Ariel Felner.

References

Li, J.; Chen, Z.; Harabor, D.; Stuckey, P. J.; and Koenig, S. 2022. MAPF-LNS2: Fast Repairing for Multi-Agent Path Finding via Large Neighborhood Search. In *AAAI*, 10256–10265.

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.