

Peer Review

Review of: "Enhancing Lifelong Multi-Agent Path Finding with Cache Mechanism"

Evgenii Arsentev¹

1. Independent researcher

1. What the work claims

The paper adds a cache to a warehouse. Not a data cache: a set of physical grid cells, drawn purple in Figure 1, placed within ± 2 rows of the unloading ports, where an agent can drop items it has fetched from a distant shelf so that the next agent needing the same item type does not have to make the same trip. The authors call the resulting system Lifelong MAPF with Cache Mechanism (L-MAPF-CM). Around the new cell type, they build a task assigner (TA) outside the path planner, a six-state machine (Figure 2: SF_GET, CA_MOV, CA_GET, CA_ADD, SF_ADD, UP_END), and a reader/writer lock per cache grid so two agents cannot rewrite a cell while a third is on its way to read it. Section IV-C3 argues the protocol cannot deadlock: an agent that fails to acquire a lock never waits; it re-routes to the shelf or straight to the port, breaking Hold and Wait. A second argument shows the state machine has no cycle avoiding UP_END.

The experimental claim (§V and the abstract) is that L-MAPF-CM beats a no-cache L-MAPF baseline in most settings, both using LaCAM as the low-level planner, and that the win tracks the cache hit rate and shrinks as congestion rises. Section V-A gives the setup: a 27×71 warehouse map adapted from the MAPF benchmark, 1,600 shelf grids, up to 80 cache grids, 4 unloading ports, agent counts from 4 to 256, cache counts in {16, 32, 48, 64, 80}, carrying capacity $P = 100$, three task distributions (MK with $M = 200$ and K in {1, 20, 40, 80, 120, 160}; the 7:2:1 "Zhang" distribution; and one built from a Kaggle product-demand dataset), and three replacement policies (LRU, FIFO, RANDOM) against NONE.

Several things here are done well, and I want to name them before I object. The cache-plus-lock framing is a good idea, and the architecture analogy is carried through rather than merely invoked: the paper works out what eviction means when eviction costs a physical round trip (CA_MOV then SF_ADD), the

part most authors would wave through. The deadlock argument in §IV-C3 is short and correct in its own terms. The failure analysis is honest: §V-B does not hide the throughput collapse at 256 agents; it points at it and backs the explanation with the wait-frequency heat map in Figure 5. The negative result about replacement policies is reported rather than buried. My objections are almost entirely on the evaluation side, and the largest is that I cannot check anything the paper says because it contains no numbers.

2. Whether the conclusion holds on the data

2.1 There is not a single measured value in the text. As rendered on Qeios, this article has no tables. Every quantitative result lives inside Figures 4, 6, and 7 as bar charts and line overlays, and those bitmaps carry no legible axis values. The running text never rescues this. The abstract says L-MAPF-CM "has demonstrated performance improvements particularly with high cache hit rates and smooth traffic conditions"; §V-B says it "surpasses the baseline in most test settings"; §VI repeats the same phrasing. I could not find one throughput value, one hit rate, one percentage, or one ratio anywhere in the body. A reader cannot tell whether the effect is 2% or 40%, cannot tell how many settings "most" covers, and cannot compare this system to anything published later. For a paper whose whole contribution is empirical, that is the defect to fix first. A results table would cost one page and change the paper's standing.

2.2 The averaging hides the factor that matters. Section V-A ends with: "All data shown in Figures 4-7 represent average values across all variables not displayed on the figures." So a bar in Figure 4 is a mean over cache counts, over replacement policies, and over the six MK values of K . Those are not repeated trials but different conditions, and one is degenerate: $K = 1$ means any window of 200 tasks holds a single item type, which any cache of any size serves at a near-perfect hit rate. Pooling $K = 1$ with $K = 160$ produces a number that describes no warehouse. The distributions are kept separate in Figures 6 and 7, which is right; K should be too.

2.3 There are no repetitions, no variance, and no test. Section V-A says agent start locations and the task queue Q are randomly generated, but not how many times. I read the section three times looking for a count of trials or seeds and found none, so each bar in Figures 4, 6, and 7 may rest on a single random draw. No caption describes error bars, no standard deviation is quoted, and no significance test appears anywhere. With a stochastic task stream, random start positions, and a randomized low-level planner, single-draw bars cannot support "surpasses the baseline in most test settings." Five to ten seeds per cell, with the spread drawn, is the minimum here.

2.4 The baseline uses random storage, which is the weakest comparator the paper itself cites. Section V-A: "Because each shelf grid represents a unique kind of item, we randomly assign an index to each shelf grid." Section III-C had already summarized three storage policies from the literature, including "[t]he turnover-based storage policy assigns items to storage locations based on their demand frequency[14]." Under the Zhang distribution, 10% of item kinds carry 70% of the demand. Putting that 10% on the shelves nearest the ports is free: no new cell type, no task assigner, no locks, no eviction traffic, and it would capture much of the travel-time saving the cache is credited with. The paper does not run it, so the reported gain is a gain over random placement, and the interesting question (does a dynamic cache beat static turnover placement, and beat the two combined?) is untouched. This is the central missing experiment. This is the central missing experiment, not a nice-to-have.

2.5 Cache warm-up is confounded with agent count. Section V-A: "We will continue generating tasks until 1,000 tasks have been completed." The caches start empty (§IV-D1: "Since the caches are empty at the beginning, agents can only retrieve items from the shelves"). The horizon is therefore fixed at 1,000 tasks while the agent count varies by a factor of 64, from 4 to 256. That gives $1000/4 = 250$ completed tasks per agent at the low end and $1000/256 \approx 3.9$ at the high end. At 256 agents, almost every agent finishes the run having done about four tasks, so the measurement is dominated by the cold-cache transient, during which L-MAPF-CM is strictly worse than the baseline: it pays the SF_GET to CA_ADD detour to fill cells whose contents it will barely reuse. The paper blames the vanishing benefit at high agent counts entirely on congestion (§V-B, Figure 5). Congestion is clearly present, but nothing in the experiment separates the two explanations, and the cold-start one follows from the stopping rule.

2.6 The MK distribution was chosen to make LRU win, and it did not. Section V-A motivates MK by citing [39], "where LRU has been proven to have an upper bound on the cache miss rate and to be better than FIFO". Section V-B then reports that "the cache replacement policy won't affect cache hit rate and final throughput performance too much". That is a contradiction between the design rationale and the outcome, and the paper lets it pass without a sentence of analysis. A possible reason follows from in-paper quantities. Each cache grid is limited to one item type (§IV-A, assumption 2), so at most 80 types are resident at any moment. At $K = 160$, if the types inside a 200-task window were roughly equally likely, no policy could exceed a hit rate of $80/160 = 50\%$, and at a miss rate that high the three policies compress together. Whether that is the real explanation is the authors' to determine, but the tension deserves a paragraph.

2.7 The two systems may not be carrying the same load. An L-MAPF-CM agent picks up as many items as it can hold (§IV-D2, `SF_GET` with a write lock: "it takes as many items as it can carry"), with $P = 100$ per §V-A. A baseline agent has no cache to deposit into, so it delivers one item per shelf trip. That extra capability is inseparable from the cache in the comparison. It may be unavoidable given the problem definition, but it should be flagged, and a cheap ablation isolates it: sweep P . If the gain is flat in P , the cache is doing the work; if it scales, batching is. I also could not determine what the purple cells become in baseline runs, aisle or obstacle. The two maps would then differ in free space, which moves congestion, the paper's own failure mode.

2.8 The 10-second budget is ambiguous and may be the binding constraint. Section V-A: "We allocate a total of 10 seconds for the L-MAPF solver to find a collision-free solution for all 1,000 tasks." Read literally, that is ten seconds of wall clock for a whole 1,000-task episode with up to 256 agents and replanning at every arrival event. I cannot tell whether the budget is per call or per episode, and the difference is large. It also matters asymmetrically: L-MAPF-CM triggers more target reassignments per task because one task can involve `SF_GET`, `CA_ADD`, and `UP_END` rather than a shelf trip and a port trip, and §IV-D2 also lets an in-flight `SF_GET` agent be re-targeted. More replanning events against a shared budget mean less search per event, so if the cap binds, the comparison is partly one of planner compute rather than route quality. The claim in the same paragraph that "the runtime overhead of L-MAPF-CM is negligible" is asserted, not measured.

2.9 Eviction traffic is never accounted for. The state machine charges eviction to an agent that was on its way to do a task: in case (c) of the `UP_END` transition, an agent with no readable and no empty cache takes a write lock, goes to a cache in `CA_MOV`, carries the evicted items to the shelf in `SF_ADD`, and only then starts its own. With up to $P = 100$ items per agent and the rule that "[i]tems evicted from cache should be sent back to shelves" (§IV-A, assumption 3), an eviction can be a long errand. The fraction of agent-timesteps spent in `CA_MOV` and `SF_ADD` is the overhead the cache must beat, and it is reported for no configuration. It is also the natural explanation for the settings where L-MAPF-CM loses, which §V-B's "in most test settings" implies exist.

2.10 What I could verify. The arithmetic of the setup is internally consistent, which is worth saying. Section V-A's 80 cache grids against 1,600 shelf grids gives $80/1600 = 5\%$, matching the stated "maximum cache-to-shelf ratio is 5%". The multi-port split holds: 4 groups at 20 cache grids each is $4 \times 20 = 80$, and the per-group agent set $\{1, 2, 4, 8, 16, 32, 48, 64\}$ scaled by 4 gives the stated totals $\{4, 8, 16, 32, 64, 128, 192, 256\}$. The Zhang masses sum correctly: $10 + 20 + 70 = 100$.

2.11 One gap in the correctness argument. Section IV-C3 concludes starvation-freedom from "each agent completes its operation in a bounded number of steps without infinite waiting for the resources". That bound belongs to the low-level planner, not to the lock protocol, and Figure 5 shows LaCAM leaving agents waiting in place under the Zhang distribution with 256 agents. The protocol is starvation-free *conditional* on the planner reaching targets in finite time, and the condition should be written down. Separately, §IV-D2 lets an `SE_GET` agent be re-targeted mid-trip whenever a readable cache appears, which is the oscillation the lock was introduced to prevent (§IV-B: agents "mov[ing] back and forth between cache and shelf grids"). Nothing in §IV-C3 rules out repeated re-targeting with no progress, and the acyclicity proof does not cover livelock.

3. Reproducibility

Some of this is good. The hardware is fully specified in §V: Ubuntu 20.04.1, AMD Ryzen 3990X 64-core, 64GB RAM at 2133 MHz. The language is given (C++), the low-level planner is named, the map is identified as an adaptation of the warehouse map from the MAPF benchmark [2], and the real-data source carries a URL in footnote 2. That is more than many papers here provide. The gaps below are still bad enough that I do not believe an independent group could reproduce Figures 4, 6, or 7.

No code location. Footnote 1 reads "Our code and video samples are available at supplementary material." On the Qeios version I reviewed, I could not find a supplementary attachment, and no repository URL, commit hash, tag, or license appears in the text. A footnote pointing at an artifact I cannot open is not an artifact.

Planner version unpinned. Section V builds on "parts of the existing LaCAM codebase[6]", where [6] is the AAAI 2023 LaCAM paper, while [29] in the same list is "Engineering LaCAM*". Which variant, at which commit, with which settings, is not stated. Its behavior at 256 agents is exactly what the congestion analysis in §V-B turns on, so this needs a commit hash. Compiler and optimization flags are absent too, and they matter under a wall-clock budget.

Seeds and trial counts. Not reported. Neither the number of random instances per configuration nor the seeds themselves appear.

Cache grid capacity is never given. This is the one parameter a cache paper cannot omit, and I looked hard. Section IV-A, assumption 2 reads "Each cache grid has the same capacity of agents and also are limited to store one type of items", which I could not parse into a number: capacity in items, or a limit on how many agents may occupy the cell? Section IV-C1 makes it load-bearing by setting the read lock's

share count to "the number of items stored in the cache". With $P = 100$ and an agent storing all but one of its items (§IV-D2, CA_ADD), a per-cell capacity of 99 is implied in one reading and contradicted in another. Every hit-rate curve depends on this number.

The real-data distribution is underspecified. Footnote 2 gives the Kaggle dataset, but not a version or download date, and §V-A says only that the authors "build a probability distribution based on the frequency of data and generate tasks from this probability distribution". How many distinct products does the dataset contain, how were they mapped onto 1,600 shelf indices, were rare products truncated or pooled, and was the temporal structure of the demand series preserved? The last question alone would move the hit rate a long way, since caches live on temporal locality. One of the three headline distributions is not reconstructible as written.

Task generation details. The MK generator is given by its defining property (at most K kinds in any window of M) but not by an algorithm, and many generators satisfy that bound with very different locality. Agent start locations are "randomly generated" with no distribution stated. Whether the 1,000-task count is per group or global in the multi-port setting is not said, and with 4 groups, the two readings differ by a factor of four.

4. What specifically to fix

1. **Add a results table.** Throughput and cache hit rate per distribution, agent count, cache count, and replacement policy, with mean and spread over seeds. This is the single change that most improves the paper.
2. **Run the turnover-based baseline from §III-C, reference [14].** Replace the random shelf indexing of §V-A with demand-frequency placement and rerun the Zhang and RDD distributions, reporting three arms: random storage without cache, turnover storage without cache, and L-MAPF-CM. If the cache still wins against turnover placement, that belongs in the abstract; if it only wins against random placement, §VI must be narrowed.
3. **Stop averaging over K .** Figures 4, 6, and 7 currently pool $K = 1$ with $K = 160$ (§V-A, "average values across all variables not displayed"), and $K = 1$ is a degenerate near-100%-hit case. Break the MK results out by K , as the paper already does by distribution.
4. **Section V-A: state the number of random trials per configuration and the seeds, and draw error bars in Figures 4, 6, and 7.** Where the paper claims L-MAPF-CM "surpasses the baseline," attach a paired test across seeds.

5. Fix the warm-up confound. Either measure over a fixed timestep horizon instead of 1,000 completed tasks, or discard a warm-up prefix, and add one figure of throughput against time at 4, 64, and 256 agents so a reader can see where the cache turns profitable. The current protocol gives roughly $1000/256 \approx 3.9$ tasks per agent at the top end.
6. §IV-A, **assumption 2**: give the per-cache-grid item capacity as a number, and restate the read lock's share count in §IV-C1 in terms of it.
7. Report eviction overhead: agent-timesteps in `CA_MOV` and `SF_ADD`, plus evictions per 100 completed tasks, for the Zhang and RDD distributions at 64 and 256 agents. Section V-B's account of the shrinking advantage is incomplete without it.
8. Clarify the solver budget in §V-A: per replanning call or per 1,000-task episode, what happens on exhaustion, whether both systems got identical budgets, and a measured TA-against-planner time to support "the runtime overhead of L-MAPF-CM is negligible."
9. Section V-A should say what the purple cells become in baseline runs, passable aisle or obstacle; otherwise, the two systems may plan on different free-space topologies.
10. Add a *P* sweep (fixed at 100 in §V-A) to separate the batching effect from the cache effect.
11. Section IV-C3: make the starvation-freedom claim conditional on the low-level planner reaching targets in bounded time, and add a short argument that the `SF_GET` re-targeting rule of §IV-D2 cannot livelock an agent.
12. Discuss the LRU/FIFO null result in §V-B rather than stating it in one clause. MK was introduced in §V-A because [39] predicts LRU should dominate FIFO there, and the prediction failed; explaining why (cache size against *K*, eviction cost, or the pooling in point 3) is a real contribution.
13. Replace footnote 1 with a repository URL, a commit hash, and a license, and pin the LaCAM variant and commit in §V.
14. Specify the RDD pipeline in §V-A: dataset version and access date, product count, the mapping onto 1,600 shelf indices, and whether temporal ordering was preserved.
15. Section numbering is broken as rendered on Qeios: §IV-C shows "2. Lock Acquisition and Release" with no visible 1, and "3. Deadlock/Starvation Free and Correctness of State Machine" lands after §IV-D's "1) Overview" and "2) State Machine." The cross-reference opening §IV-D1 ("As shown in IV-C2") evidently means Algorithm 1.
16. Text corrections: "senerio" in §IV-A; "When `CA_MOV` agents arrive caches" in §IV-D2 is missing "at"; "Their target are then updated" needs "targets"; and near the end of §V-B, a full stop runs into the next sentence without a space ("overly simplistic for L-MAPF-CM.Performance could also").

5. Recommendation

Major revisions.

The mechanism here deserves better than the evaluation it was given. As far as I can tell from §III-C, the positioning is fair: prior cache-in-warehouse work sits at the layout and queueing level, and this is the first treatment I know of that puts cache cells into the planning graph and then deals with the consequences, eviction as physical travel and races between agents included. The lock design is sensible, the state machine small enough to reason about, and §IV-C3 does reason about it. The paper is candid: it reports that the replacement policy barely matters, shows its worst case in Figure 5, and says plainly that the benefit shrinks as agents are added.

What stops me from recommending acceptance is that the empirical claims are unverifiable as the paper stands. No numeric result appears in the text; the figures pool a degenerate condition ($K = 1$) with hard ones; nothing indicates any configuration was run more than once; no statistical claim is made; and the only baseline is a no-cache system over randomly indexed shelves, when §III-C itself describes turnover-based placement as established practice. Together, these mean that "L-MAPF-CM has demonstrated performance improvements" rests on bar heights I can neither read nor recompute.

None of this needs new ideas, and none of it invalidates the mechanism. Points 1 through 5 are the ones I would treat as conditions: a results table, the turnover baseline, unpooled K , repeated seeds with spread, and a fix for the warm-up confound. Points 6 through 10 close specification holes that block reproduction. If those come back and the cache still wins under the Zhang and real distributions at moderate agent counts, this becomes a solid paper with a citable result, and I would expect to recommend acceptance. I would also keep the negative findings prominent. That the advantage evaporates under congestion at 256 agents is more useful to the warehouse community than another win, and it points straight at the corridor and guidance-graph work §V-B already gestures toward in references [41] and [42].

Declarations

Potential competing interests: No potential competing interests to declare.