Lifelong MAPF
in context of League of Robot Runners

Seminar on Artificial Intelligence 2, Prague 2026

Vojtěch Dvořák

dvorak@ufal.mff.cuni.cz

Lifelong MAPF

MAPF
- Graph of vertices and edges.
- Each agent has start and goal vertex.
- Agent moves to an adjacent vertex or waits.
- Minimizing sum of costs.

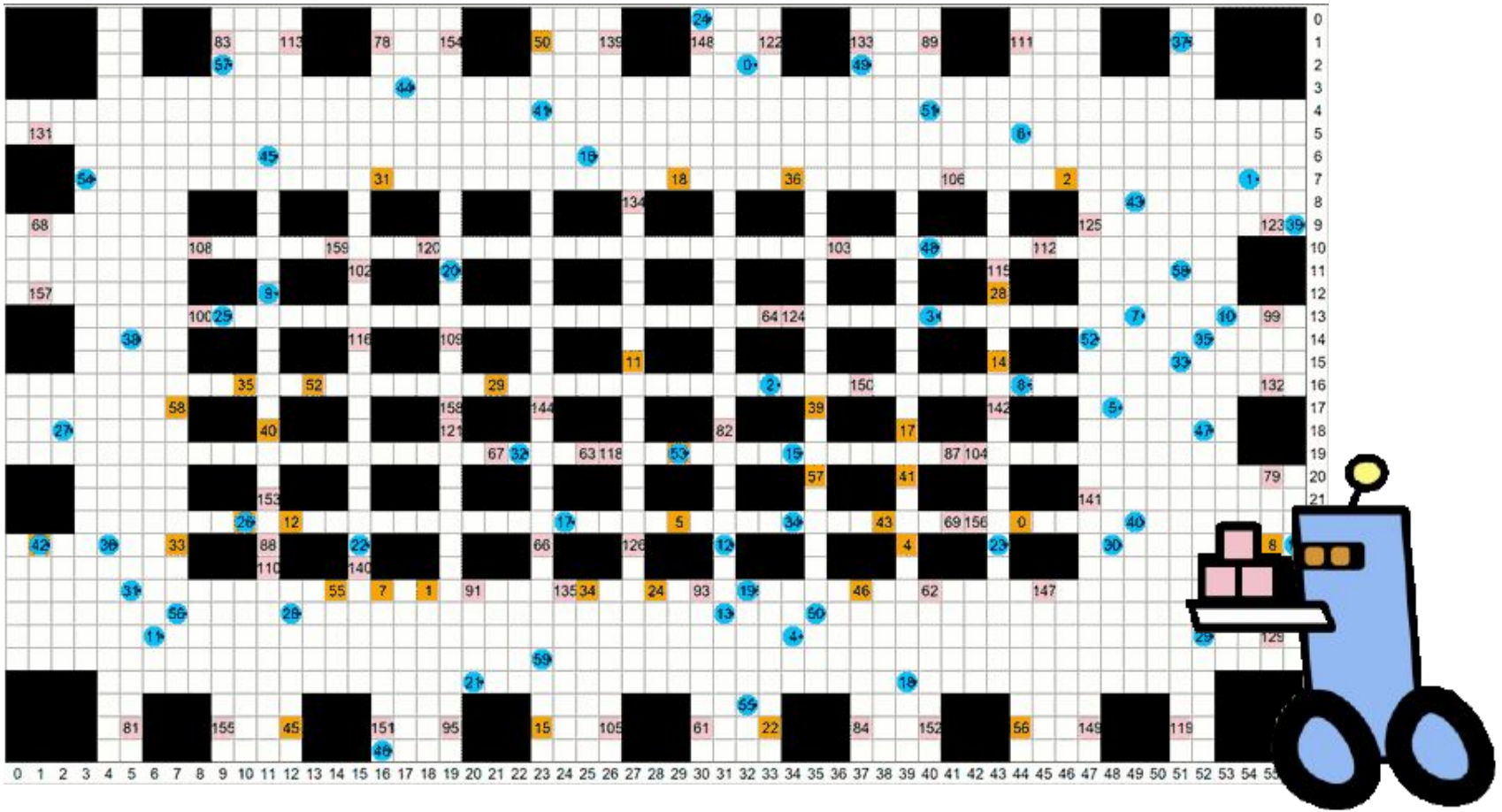
Lifelong MAPF (LMAPF)
- Agents are assigned **new tasks** constantly.

MAPF with rotations (MAPF-R)
- Graph is a grid map.
- State of an agent is its location and **rotation**.
- Agent moves to an adjacent vertex, waits or **rotates by 90°**.

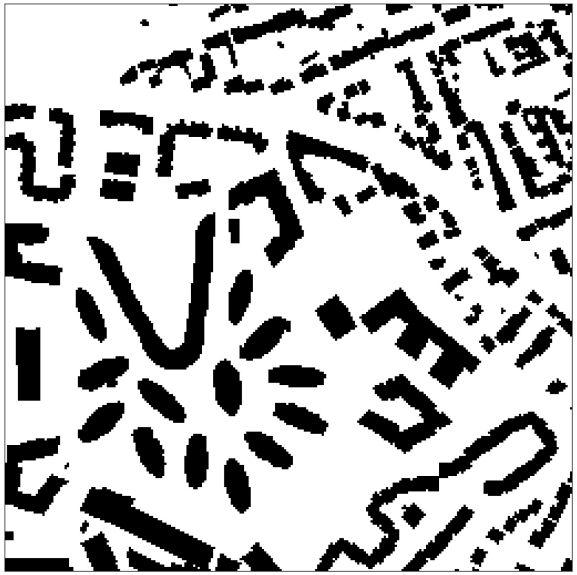
Motivation:
How Amazon robots navigate congestion

League of Robot Runners (LRR)

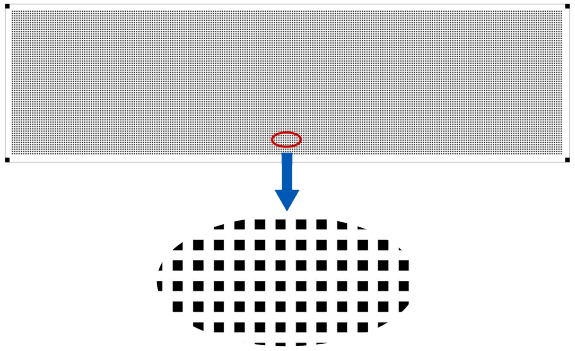
- Organized by Amazon since 2023.
- Combination of MAPF-R and LMAPF.
- 30 minutes for preprocessing, than 1 second for each step.



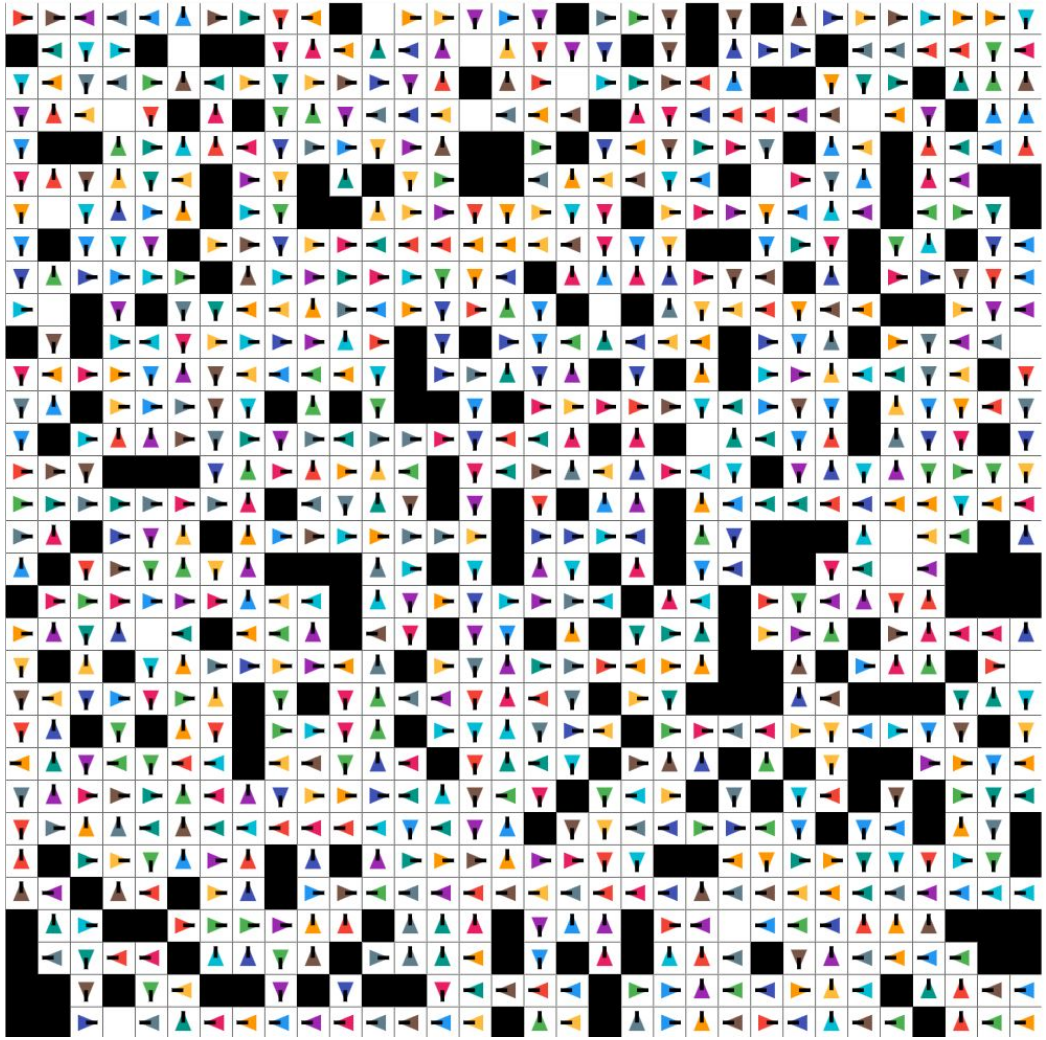
LRR Maps



City



Sortation



Random

Difference between LRR and other benchmarks

Stern et al. (2019)
- MAPF
- Agent count: **≤ 1,000**
- Agent density*: **≤ 50 %**
- Planning limit: **1-5 minutes**

Chen et al. (2024)
- LMPAF
- Agent count: **> 10,000**
- Large maps

LRR (2023)
- LMAPF-R
- Agent count: **≤ 10,000**
- Agent density*: **≤ 97.7 %**
- Planning limit: **1 second**
- Large maps

* *Agent density* is defined as number of agents divided by the number of vertices.

LRR 2026

- Similar to LRR 2023.
- Introduced delay probabilities.

Scaling Lifelong Multi-Agent Path Finding to More Realistic Settings: Research Challenges and Opportunities

He Jiang, Yulun Zhang, Rishi Veerapaneni, Jiaoyang Li

2024

Paper overview

- First place in LRR 2023.
- Score **9.755** (second team 9.502).
- Second and third team based their solutions on A* and CBS (Conflict Based Search).

Planning

- Key challenges: need for **Anytime** and **Parallel** algorithm.
- Initial solution from **PIBT** (Priority Inheritance with Backtracking), without collisions, takes at most 250 ms.
- Improvements made with **MAPF-LNS** (MAPF Large Neighborhood Search).
- ⇒ **PIBT-LNS**.
- Running in parallel and with sliding window (**WPPL**).

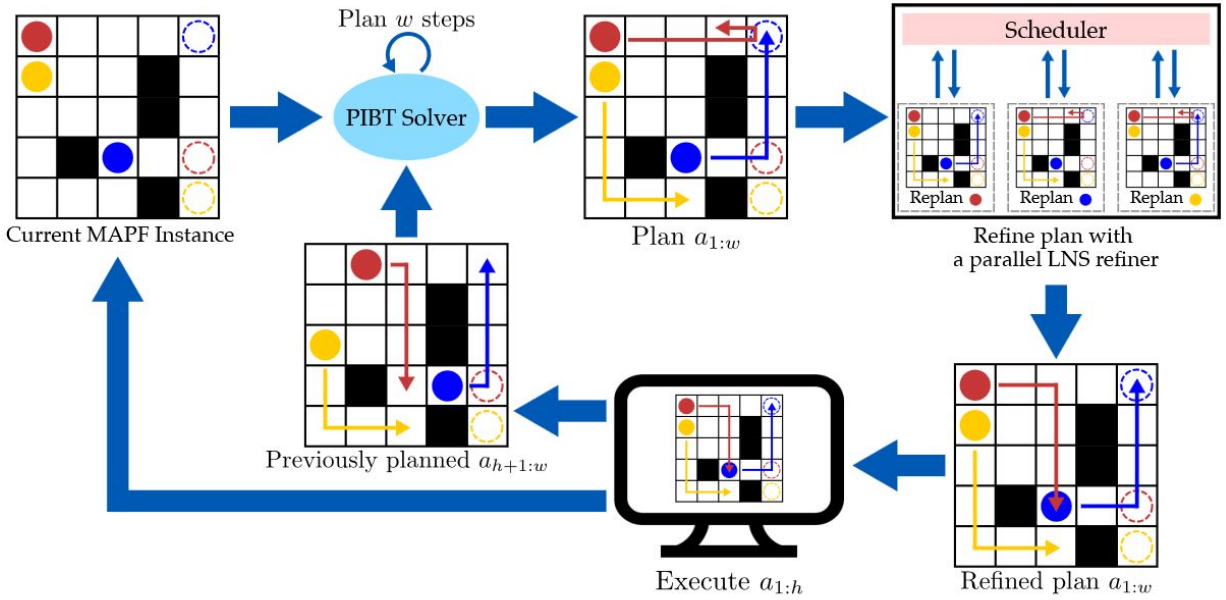
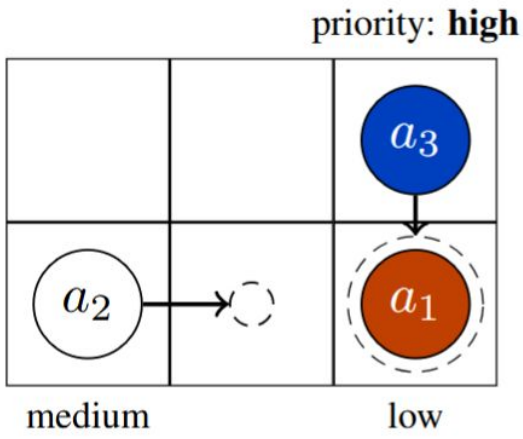


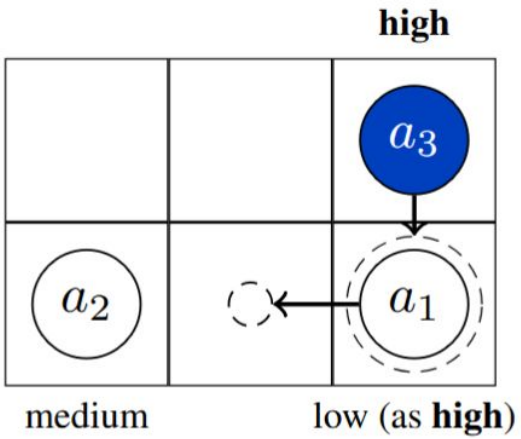
Figure 2: Windowed Parallel PIBT-LNS (WPPL). For each windowed MAPF instance, we run PIBT w steps to get an initial plan $a_{1:w}$ and use Parallel MAPF-LNS to refine it. We then execute the first h actions of the refined plan $a_{1:h}$, updating the MAPF instance and reusing the rest of actions $a_{h+1:w}$ in the next iteration of PIBT.

Overview of PIBT

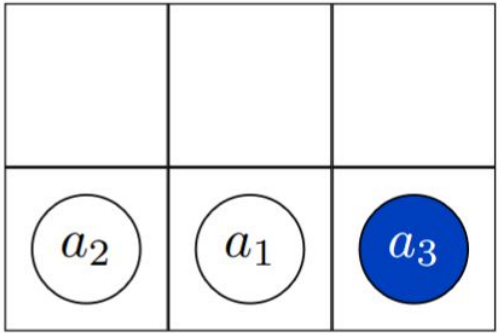
- Agents are assigned priorities that can be inherited.
- Plans only one step ahead.
- “Pushing” agents that are in way.



(a) Stuck agent

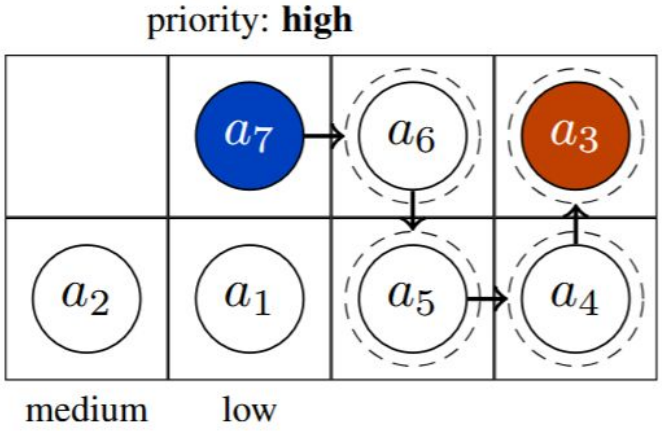


(b) Priority inheritance

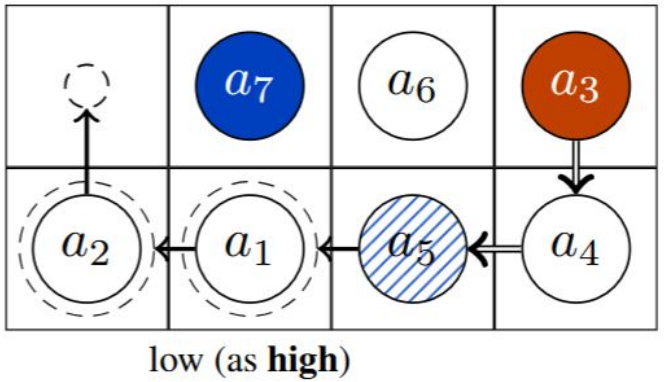


(c) One timestep later

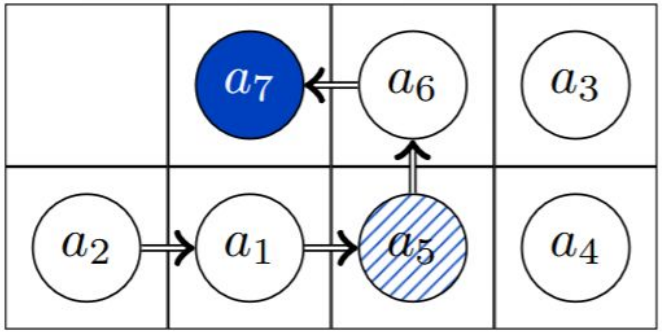
Overview of PIBT



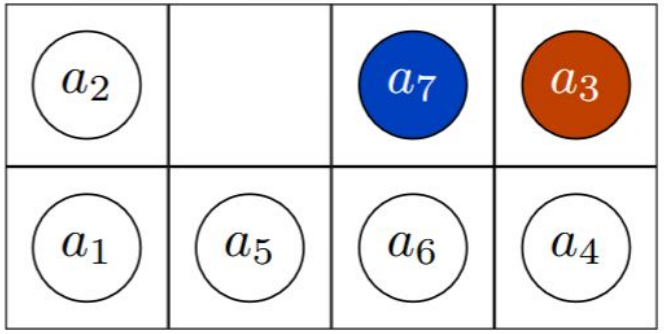
(a) Priority inheritance



(b) Backtracking and priority inheritance again



(c) Backtracking



(d) one timestep later

Overview of MAPF-LNS

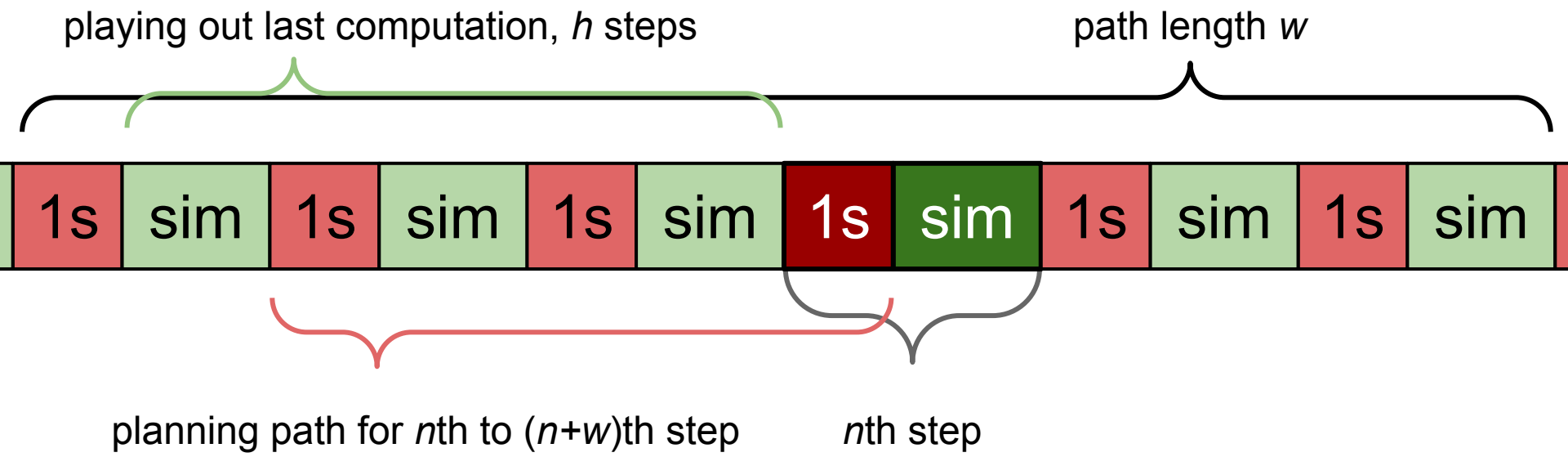
- **Destroy:**
  - Select a small “neighborhood” of *k* agents.
  - And remove their paths from the solution.
- **Repair:**
  - Replan those *k* agents (using cheap planner i. e. A*).
  - Other agents' paths are obstacles.
  - ⇒ New collision-free sub-plan.
- **Accept or reject:**
  - Keep the new plan if it reduces the total cost
  - Otherwise discard it and try a different neighborhood.

- **Neighborhood selection**:
  - Random.
  - Largest detour.
  - Map topology (collect all agents that visit chosen intersection).

Sliding window

- Planning window of length *w*.
- Replanning agents only every *h* steps, *w* ≥ *h*.

- Example: *h* = 3, *w* = 6.



Hyper-parameter overview

- Planning window of length *w*.
- Replanning agents only every *h* steps, *w* ≥ *h*.

Instance	Algorithm	w	h	GG	DA	Score
Random 100	WPPL	20	1	No	No	0.994
Random 200	WPPL	15	1	Manual	No	0.975
Random 400	WPPL	15	1	Manual	No	0.992
Random 600	WPPL	15	1	Manual	Yes	1.000
Random 800	PIBT	1	1	GGO	Yes	0.806
City 1000	WPPL	20	1	No	No	0.996
City 3000	WPPL	20	1	Manual	No	1.000
Game 4000	WPPL	15	3	Manual	Yes	1.000
Warehouse 8000	WPPL	10	3	Manual	No	0.996
Sortation 10000	WPPL	10	3	Manual	No	0.997

large maps

Instance name is a map name followed by number of individual agents.

Improvements to WPPL

Traffic congestion

- Myopic behaviour.
- Manual and automatic design of guidance graphs.
  - Assign weight to every action at every vertex.
  - Optimization is expensive.

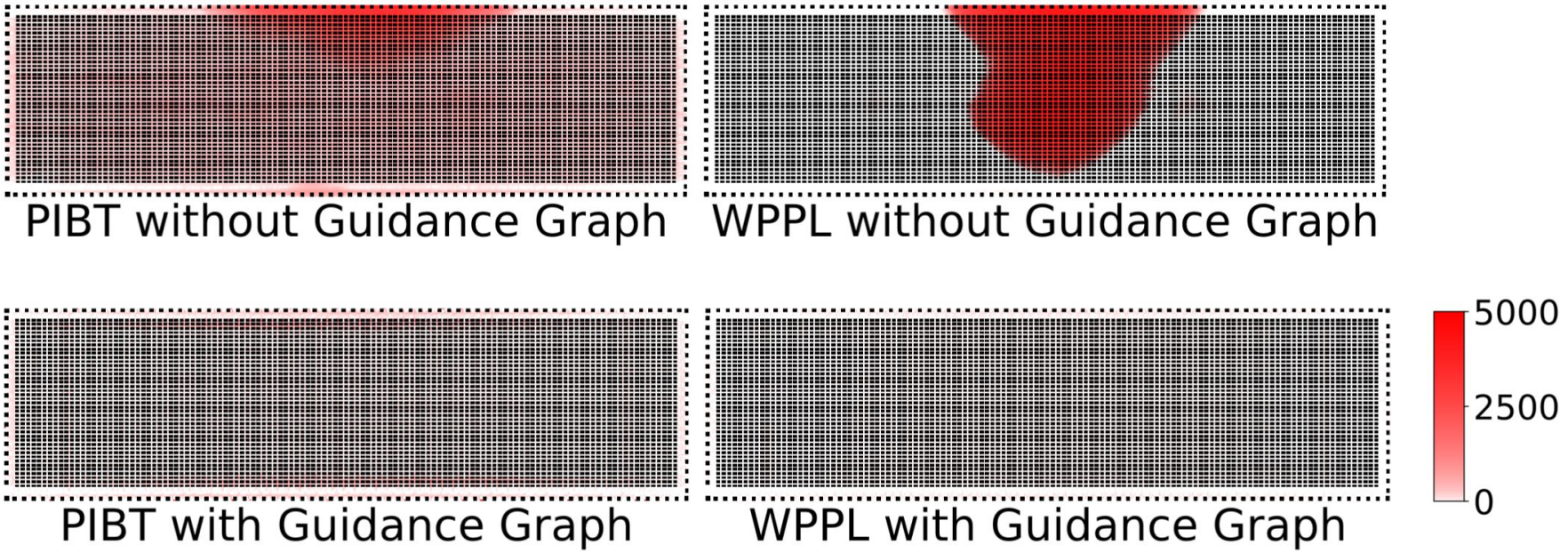
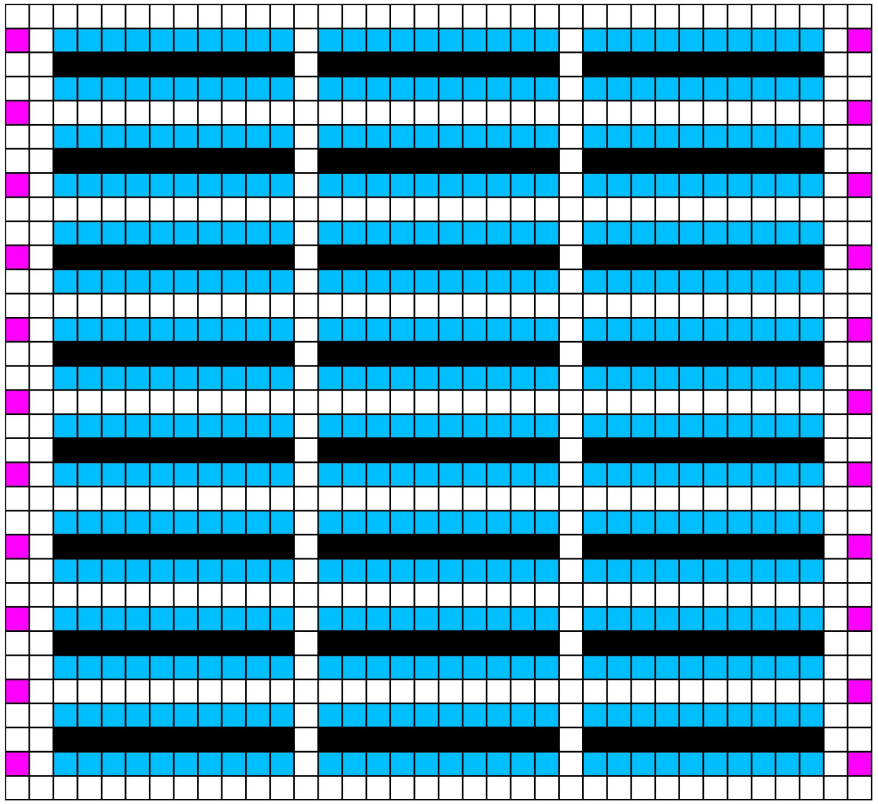
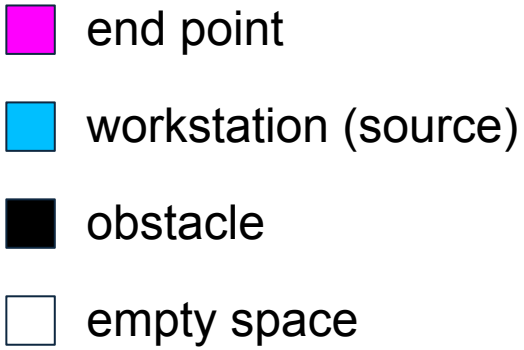


Figure 5: Comparison of with and without guidance graph in the Warehouse 8000 instance. The heatmap shows the wait action usage (the number of steps agents wait in each vertex). Red denotes areas of high congestion.

Guidance graphs example

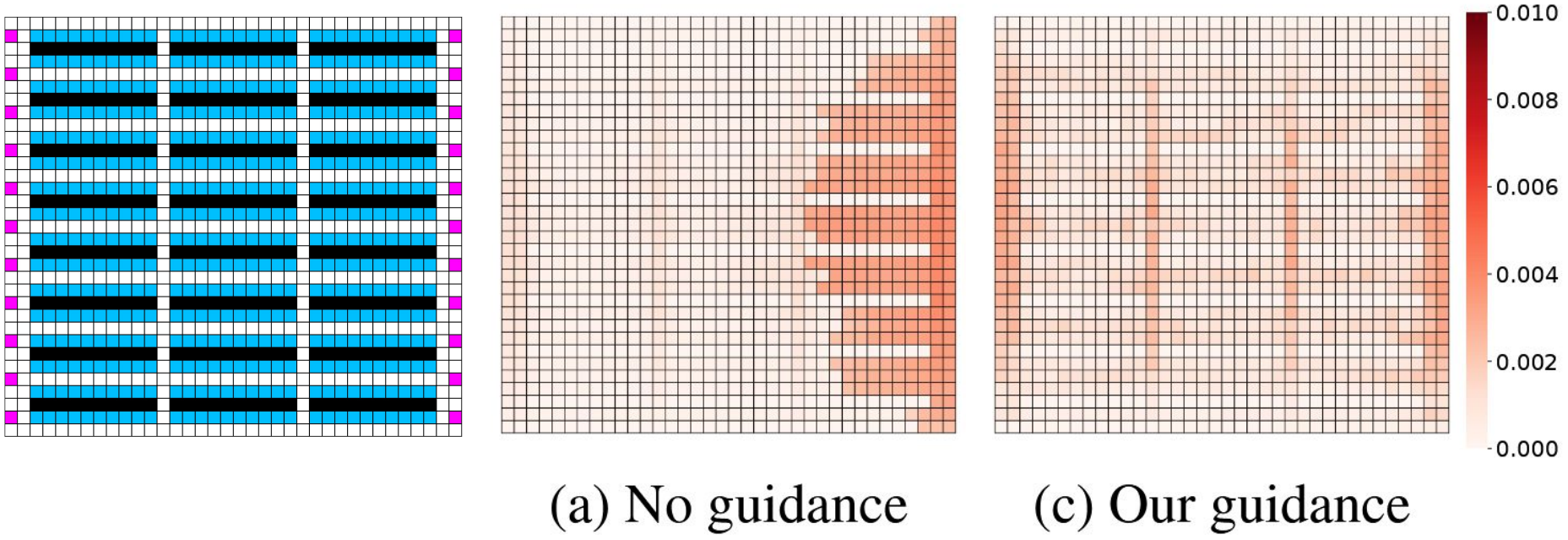
- Guidance Graph Optimization for LMAPF, Yulun Zhang et al., 2024





Guidance graphs example

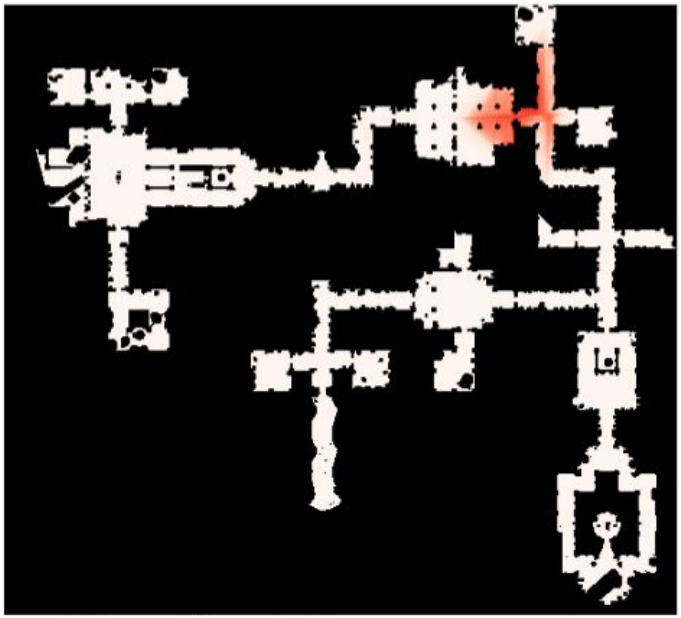
- Guidance Graph Optimization for LMAPF, Yulun Zhang et al., 2024



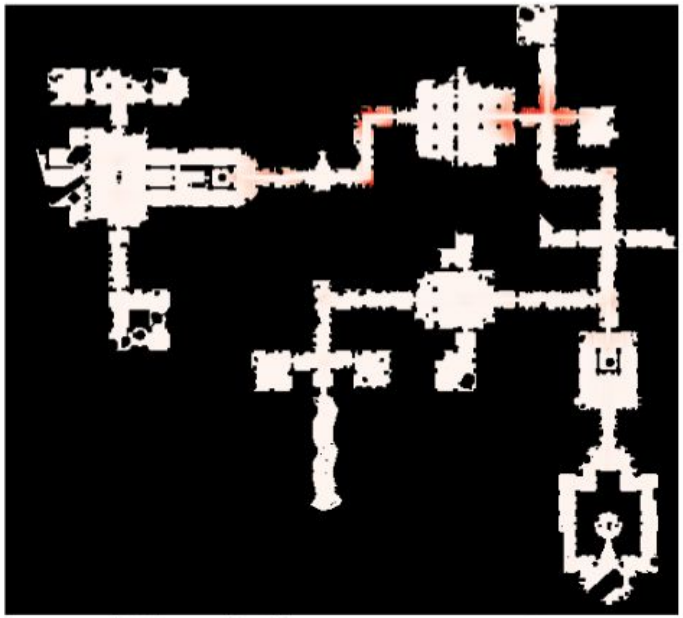
The heatmaps show the tile-usage (the frequency that each tile is occupied).

Disabling agents

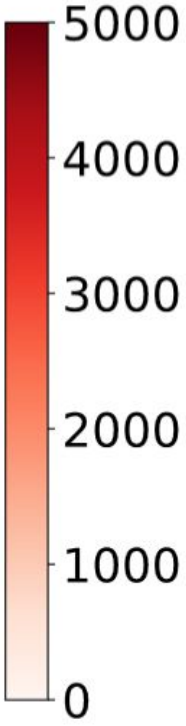
- Randomly choose agents “to make way” if needed.
- For *Game 4000* disabling 1500 agents leads to improvement.



No disabling agents

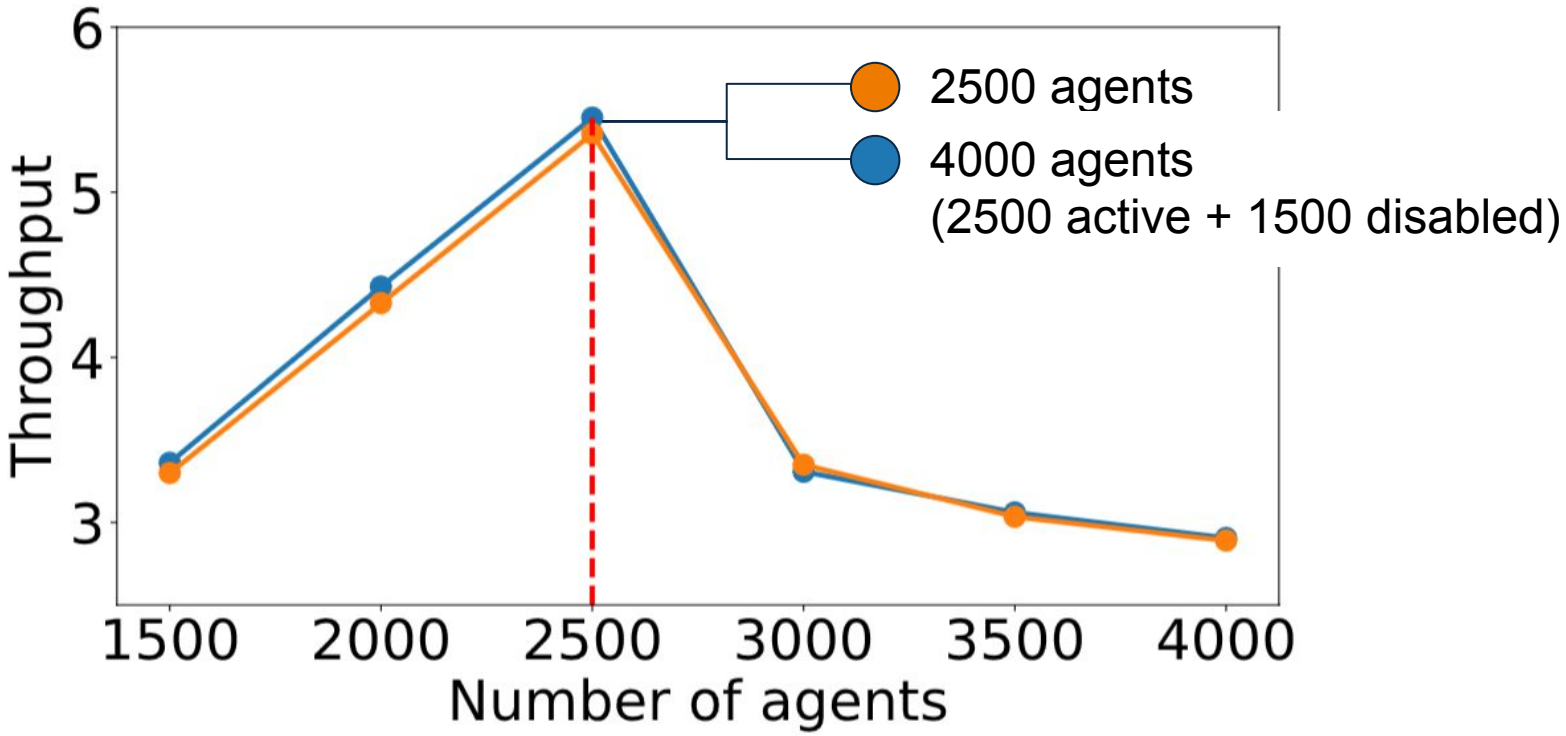


Disabling agents



Disabling agents

- Randomly choose agents “to make way” if needed.
- For *Game 4000* disabling 1500 agents leads to improvement.

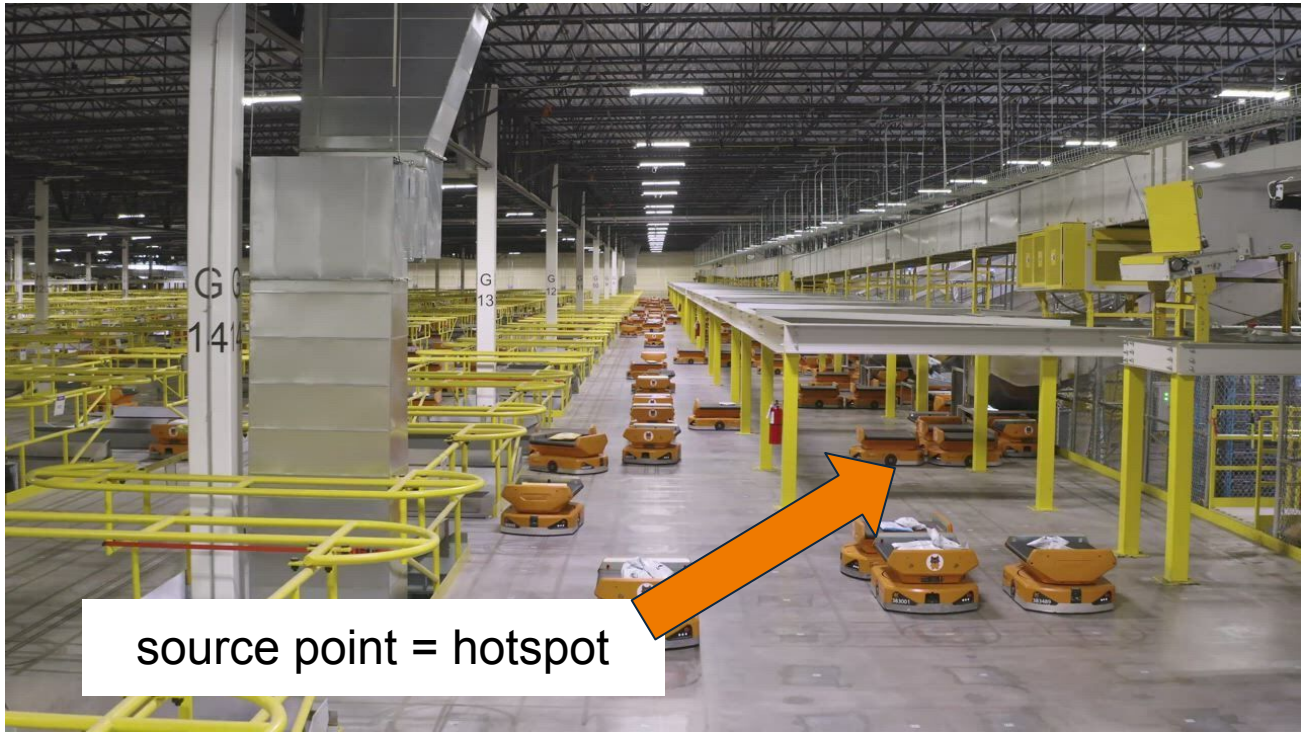


Possible future improvements

- Exploiting parallelisation.
- Better heuristics for neighborhood selection for MAPF-LNS.
- Search for optimal number of agents.
- Faster automatic guidance graph design.

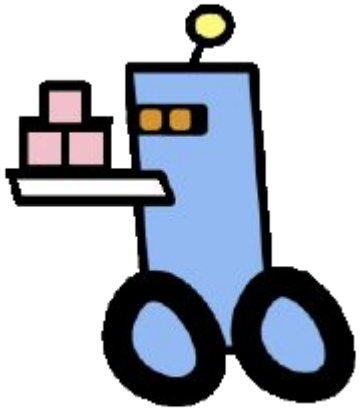
Modeling MAPF vs Reality

- Kinematics and dynamics.
- Uncertainty during execution (tackled by LRR 2026).
- Non-uniform task distribution.
- Changes in map layout and agent numbers.



Thanks for your attention

QA



References

1. Jiang, H., Zhang, Y., Veerapaneni, R., & Li, J. (2024, June). Scaling lifelong multi-agent path finding to more realistic settings: Research challenges and opportunities. In *Proceedings of the International Symposium on Combinatorial Search* (Vol. 17, pp. 234-242).
2. Okumura, K., Machida, M., Défago, X., & Tamura, Y. (2022). Priority inheritance with backtracking for iterative multi-agent path finding. *Artificial Intelligence*, 310, 103752.
3. Li, J., Chen, Z., Harabor, D., Stuckey, P. J., & Koenig, S. (2021). Anytime multi-agent path finding via large neighborhood search. In I*nternational joint conference on artificial intelligence 2021* (pp. 4127-4135). Association for the Advancement of Artificial Intelligence (AAAI).
4. Brown, A. S. 2022. How Amazon Robots Navigate Congestion. https://www.amazon.science/latest-news/how-amazon-robots-navigate-congestion.
5. Zhang, Y., Jiang, H., Bhatt, V., Nikolaidis, S., & Li, J. (2024). Guidance graph optimization for lifelong multi-agent path finding. arXiv preprint arXiv:2402.01446.