Lifelong MAPF and Task Assignment Considering Workers in Warehouses

Naoki Mizumoto,[1] Katsuhide Fujita,[1] Yoshihiro Ueda,[2] Takayoshi Mori,[2]
[1]Tokyo University of Agriculture and Technology
[2]TOPPAN Digital Inc.

Abstract

In this study, we apply multi-agent path finding (MAPF) to actual warehouse operations. In conventional MAPF, it is typically assumed that a single type of agent discovers a route toward a single destination. However, this study focuses on the problem of two types of agents (robots and humans) working together in warehouse operations by routing to multiple destinations. We propose an approach that combines task allocation using network flow and route search using a Monte Carlo tree search and prioritized planning.

Introduction

In recent years, research on the application of multi-agent systems to various social aspects has been conducted. An essential application of multi-agent systems is the automation of picking operations in warehouses. The picking operations involve taking specified items from the shelves of a warehouse and transporting them to a specified collection point(Wurman, D' Andrea, and Mountz 2008). In the past, this task was performed by humans alone. However, the development of transport robots is gradually progressing, and robots have recently been able to pick objects(Liang et al. 2015). However, although such robots are technically feasible, their high cost makes it challenging to implement them in practical operations from a financial perspective. In addition, warehouse layouts would be modified to allow picking robots to handle items efficiently. Conversely, robots equipped solely with the ability to load and transport items, without any picking ability, can be introduced at a relatively low cost. This study focuses on scenarios in which transport-specific robots collaborate with human workers who are responsible for picking tasks.

Many existing studies aim to automate picking tasks in warehouses(Kou et al. 2020)(Ma et al. 2017)(Liu et al. 2019), but only a few have considered how robots and humans can work together to perform picking tasks. In addition, few studies have considered how to assign tasks to agents to determine which items they should be responsible for(Ma and Koenig 2016). This means that most existing studies simplify the experimental setup. Therefore, this study focuses on warehouse picking tasks while considering task allocation and multi-agent path planning while considering collisions between robots and workers.

The primary contributions of this study can be summarized as follows:

- We model the warehouse environment and define the behavior of robots and workers.
- We propose a method for task allocation to robots and workers and a path planning and task exchange algorithm based on this task allocation.
- We evaluate the effectiveness of the proposed task allocation method by evaluating the total work time of all agents, the total work time of each agent, the number of collisions, etc. as an evaluation experiment.

In our experiments, we use datasets from actual warehouse picking tasks.

Related Work

Multi-agent path finding (MAPF) is the process of determining a route to a destination while considering collisions between multiple agents(Stern et al. 2019). In particular, the MAPF problem that considers the case where multiple destinations have been set and agents move to another destination after reaching one destination is called the Lifelong MAPF problem(Li et al. 2021).

Formally, the Lifelong MAPF problem is formulated as a graph theory problem. An unweighted, undirected graph ($\mathcal{G}$) and a set of agents ($\mathcal{A}$) are given. Each agent (i) is located on a node of $\mathcal{G}$ and is given an initial position (s_i) and multiple destinations ($\boldsymbol{\tau}_i$), where $i = 1, 2, ..., n$. Each agent needs to determine a route from its initial position to its destination. Agents can move simultaneously and can move to a neighboring node, i.e., a node connected by a link, only in each 1 step. However, agents must not cause a collision, such as using the same node or link simultaneously. The Lifelong MAPF problem involves determining a route for all agents to reach all destinations under these conditions. Note that the order of the destinations of each agent, $\boldsymbol{\tau}_i = \{\tau_{i1}, \tau_{i2}, ..., ..., \tau_{iK}\}$, may be changed. In addition, when the destination (task) can be freely assigned to any agent, it is called anonymous MAPF(AMAPF).

Solutions to Lifelong MAPF include approaches that use nonlearning algorithms (Jiang et al. 2024) and PRIMAL2,

Figure 1: Condition for picking

which employs distributed reinforcement learning (Damani et al. 2021). In Lifelong MAPF, goals are assigned continuously; however, they can be addressed by adapting solutions from one-shot MAPF. However, this method does not consider human workers and assumes only the presence of robots. Furthermore, in this study, multiple items (goals) are assigned to agents, and these items must be allocated to robots, making the problem more complex than in previous research.

Problem Statement

Most of the formulations are the same as those for Lifelong MAPF demonstrated in related studies; however, some special constraints are provided.

The map information is provided as an undirected graph $\mathcal{G}$. There are two types of agents: robots and workers. Here, each agent is given an initial position (x_i, y_i). Tasks are given in units called order sets $\mathcal{T}$. An order set consists of multiple orders $(\tau_1, \tau_2, \ldots, \tau_N)$, and each order is associated with a node on the graph. A robot has B boxes for loading order sets; thus, it can process up to B order sets in a single task assignment. Orders cannot be processed by a robot alone; they need to be picked up in cooperation with a worker. To select an order, if the node number corresponding to the order is v, the robot and worker must each stay at node numbers v and $v+1$ for T s. After processing several order sets, the robot returns to the depot to unload the order sets and empty the box. By repeating this process, all order sets are processed.

Although it is technically feasible to manage the position of workers, practical implementation incurs significant costs. In addition, workers may not necessarily follow the exact path determined by the algorithm. However, it is possible to identify which shelves workers performed picking tasks during inventory management. Furthermore, even if human movements slightly deviate from the simulation, they are expected to avoid collisions and follow relatively short paths. The method proposed in this study is scalable and computationally efficient, allowing recalculations at each step. This enables rapid calibration in response to minor deviations. Therefore, it is reasonable to conduct simulations and experiments using the path planning results generated by the algorithm, as is done with robots.

Proposed Method

Task Assignment

The task assignment comprises the following components.

1. Order set assignment
2. Determining the sequence of visiting assigned order sets
3. Task assignment to workers
4. Task exchange during a route search

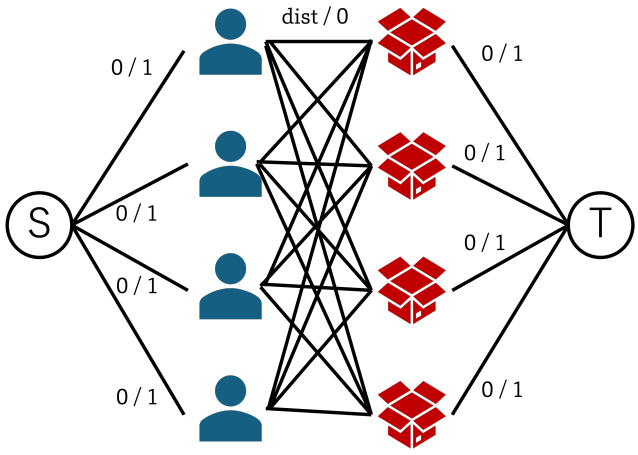

Figure 2: Network flow for matching

Order Set Assignment

The locations where tasks occur tend to be congregated. Therefore, we avoid collisions between agents by clustering order sets. Let (x_i, y_i) be the coordinates of the ith order in the order set. Then, the center of gravity of the order set $(\bar{x}, \bar{y}) = (1/N)(x_i + y_i)$, is calculated. Based on the coordinates of the center of gravity, clustering is performed using the k-means method. Then, tasks belonging to the same cluster are assigned to the same agent. Here, the clusters that each agent is in charge of are assigned to be as different as possible.

Determining Sequence of Visits for Assigned Order Set

The order set is random; thus, it is necessary to determine the appropriate sequence of visits. It is better to consider collisions in the sequence of visits; however, this approach incurs significant computational costs in practice. Therefore, we apply the results of solving the problem involving a single agent. In the case of a single agent, it is sufficient to determine the sequence to visit a set of orders on a graph, $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_{|\mathcal{T}|}\}$. This problem is well-known as the traveling salesman problem (TSP). The TSP searches the shortest route that visits all vertices exactly once and returns to the starting point at the end. The TSP is NP-hard; however, approximate and exact solutions have been proposed. In this study, we use 2-opt(cro 1958) to find an approximate solution.

Task Assignment to Workers

Unlike robots, workers only perform picking tasks with robots. Therefore, there is no need to assign tasks to workers in units called “order sets,” and tasks can be assigned one-to-one to the nodes that robots will visit at each step. In

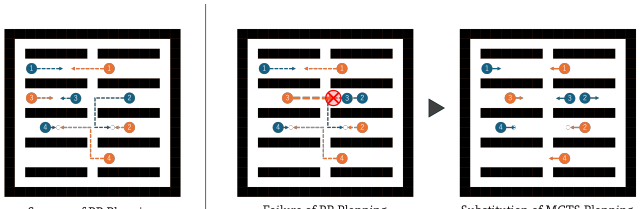

Figure 3: MCTS-PP for Multi-agent Path Finding

other words, weighted bipartite matching between the destinations of workers and robots is sufficient. This problem can be formulated and solved as a minimum cost flow problem, which is a network flow problem. A graph is created separately for the network flow from the map information. First, we prepare vertices S and T for convenience. The vertex S is connected to the destination R_i of the ith robot with an edge of capacity 1 and cost 0. Similarly, the ith worker C_i is connected to vertex T with an edge of capacity 1 and cost 0. The robot's destination R_i is connected to worker C_j with an edge of capacity 1 and cost c_ij. Here, c_ij is determined by the distance between the robot's destination and the worker. The minimum cost of the flow is calculated as much as possible with starting point S and endpoint T in this graph. The edges between the robot's destination and the worker are allocated to the worker from the flowing edges.

Task Exchange During Path Search; TE

Tasks should be allocated to avoid collisions as much as possible at the time of order set allocation. However, tasks must be allocated in units of "order sets"; therefore, it is difficult to avoid collisions. If two agents aim for the same node during a route search, negotiations are held, and the agent with the longer distance to the destination changes its destination. The destination column is arranged such that the distance between them is reduced using the TSP; thus, the exchange cost is relatively small.

Multi-agent Path Finding

Prioritized planning (PP) is a simple method for solving Lifelong MAPF(Erdmann and Lozano-Perez 1986)(Silver 2021). PP searches for a path using the following procedure.

1. Each agent is assigned a priority. The priority is determined by considering the shortest distance to the destination and the matching of robots and workers.
2. The path to the destination of the agent with the highest priority is searched. Here, collisions with the path decided by the agent with higher priority are avoided.

PP has low computational complexity and is excellent in terms of responsiveness when a solution can be found. However, PP does not guarantee the discovery of feasible solutions. Therefore, if a route cannot be found using PP, it is searched using Monte Carlo tree search (MCTS)(Browne et al. 2012). MCTS has achieved good results in games such as chess and Go(Silver et al. 2016). MCTS is also applicable to multi-agent systems, including MAPF problems(**?**)(Pitanov et al. 2023)(Skrynnik et al. 2024). MCTS constructs a search tree based on random sampling. It is important to consider which actions to search for and how to evaluate searched nodes. Each node in the Monte Carlo tree represents a state comprising the positions of the agents and the progress of the picking tasks. A parent node generates potential next-step states by reflecting the possible plans of each agent based on its current state. The generated states are then expanded as child nodes. When selecting an action, the action with the most recent future action is determined and selected using UCB1-tuned(Auer 2002)(Tak, Lanctot, and Winands 2014):

$$UCB1_{turned} = \bar{x}_a + \sqrt{\frac{\log N}{n_a} \min\left(\frac{1}{4}.\hat{x}_a + \sqrt{\frac{2\log N}{n_a}}\right)}$$

Here, $\bar{x}_a$ denotes the average reward of the child node, N denotes the total number of trials, n_a denotes the number of trials for the child node, and $\hat{x}_a$ denotes the variance of the reward of the child node. The reward obtained during back-propagation is determined by assuming that each agent has reached the destination. If the destination is reached, the reward is 1; otherwise, the reward is 0. Furthermore, if the distance between the current node and destination is less than d, the reward is 0.1. If the same action is selected θ times, a further search is performed from that action. In the expansion, all possible actions of the next agent are expanded according to the priority used in PP.

Algorithm 1 MAPF algorithm

Require: $priorities$: List of agent priorities
Ensure: $routes$: List of planned routes
1: $routes \leftarrow [\text{None}] \times N$
2: **for** i in $priorities$ **do**
3: $routes[i] \leftarrow \texttt{PPSolution}(agents[i])$ {Returns an empty array if no collision-free route exists}
4: **end for**
5: **if** `exists_empty`(routes) **then**
6: $routes \leftarrow \texttt{MCTSSolution}(priorities, agents)$
7: **end if**

Evaluation

Experimental Setup

We used one map and three order sets for the evaluation. The map had a typical warehouse structure and corresponded to 982 nodes in the graph representation. The picking task required $T = 3$ steps. We used three task sets, with each set containing 447, 481, and 826 tasks. During execution, the MCTS algorithm performed 1000 selection–expansion iterations.

- Dataset 1 includes the first task set, 5 robots, and 3 workers.
- Dataset 2 includes the first task set, 7 robots, and 7 workers.

Table 1: Makespan and AR values of MCTS-PP + TE + TSP, MCTS-PP + TSP, and MCTS-PP + TE on datasets 1 – 6

	Method	Makespan	AR
Dataset 1	MCTS-PP + TE + TSP	1995.0	0.8
	MCTS-PP + TSP	1993.2	0.2
	MCTS-PP + TE	4072.3	1
Dataset 2	MCTS-PP + TE + TSP	1968.0	0.7
	MCTS-PP + TSP	1993.0	0.2
	MCTS-PP + TE	4072.3	1
Dataset 3	MCTS-PP + TE + TSP	4002.4	1
	MCTS-PP + TSP	4036.1	1
	MCTS-PP + TE	4213.0	1
Dataset 4	MCTS-PP + TE + TSP	2435.9	0.8
	MCTS-PP + TSP	2429.3	0.3
	MCTS-PP + TE	4298.4	1
Dataset 5	MCTS-PP + TE + TSP	4002.4	1
	MCTS-PP + TSP	4036.1	1
	MCTS-PP + TE	5528.6	1
Dataset 6	MCTS-PP + TE + TSP	3167.4	0.9
	MCTS-PP + TSP	3379.5	1
	MCTS-PP + TE	4106.0	1

- Dataset 3 includes the second task set, 5 robots, and 3 workers.
- Dataset 4 includes the second task set, 7 robots, and 7 workers.
- Dataset 5 includes the third task set, five robots, and 3 workers.
- Dataset 6 includes the third task set, 7 robots, and 7 workers.

The experiments were conducted 10 times using the same dataset and methodology. The evaluation was based on the average makespan and achievement rate (AR) of completing all tasks. Here, makespan represents the number of steps required to complete all tasks.

Experimental Results

In Table 1, the makespan of MCTS-PP + TE+ TSP is significantly reduced across all datasets. This result implies that the integration of MCTS-PP + TE+ TSP provides substantial value even with a one-shot TSP approach. Although the difference between MCTS+TE and the proposed method is small, MCTS+TE achieves slightly better performance and improves the AR.

In Table 2, the makespan of the approach with clustering is reduced compared to that of the approach without clustering. However, no significant difference is observed in the AR. In addition, the appropriate value of k varies depending on the dataset.

Conclusion

We focused on the MAPF problem modeled by actual warehouse picking tasks. We proposed an approach that combines task allocation using network flow and route search via MCTS and PP. The proposed approach achieved high performance in terms of makespan and AR based on evaluations on actual datasets. In addition, the proposed approach was analyzed using ablation studies.

Table 2: Effect of varying k values in k-means method

	k	makespan	AR
dataset 1	1	1968.0	0.7
	2	2023.8	0.6
	3	2013.5	0.8
	4	1995.0	0.8
dataset 2	1	2234.2	1
	2	2049.6	0.9
	3	2064.3	1
	4	1960.0	1
dataset 3	1	2573.1	0.7
	2	2439.9	0.8
	3	2444.1	0.7
	4	2435.9	0.8
dataset 4	1	2694.5	1
	2	2474.1	1
	3	2387.8	1
	4	2375.9	1
dataset 5	1	4376.9	1
	2	4002.4	1
	3	4079.6	1
	4	4037.7	0.9
dataset 6	1	4376.9	1
	2	4002.4	1
	3	4079.6	1
	4	4037.7	0.9

References

Auer, P. 2002. Finite-time analysis of the multiarmed bandit problem.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4(1):1–43.

1958. A method for solving traveling-salesman problems. *Operations research* 6(6):791–812.

Damani, M.; Luo, Z.; Wenzel, E.; and Sartoretti, G. 2021. Primal _2: Pathfinding via reinforcement and imitation multi-agent learning-lifelong. *IEEE Robotics and Automation Letters* 6(2):2666–2673.

Erdmann, M., and Lozano-Perez, T. 1986. On multiple moving objects. In *Proceedings. 1986 IEEE International Conference on Robotics and Automation*, volume 3, 1419–1424.

Jiang, H.; Zhang, Y.; Veerapaneni, R.; and Li, J. 2024. Scaling lifelong multi-agent path finding to more realistic settings: Research challenges and opportunities. In *Proceedings of the International Symposium on Combinatorial Search*, volume 17, 234–242.

Kou, N. M.; Peng, C.; Ma, H.; Kumar, T. S.; and Koenig, S. 2020. Idle time optimization for target assignment and path

finding in sortation centers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 9925–9932.

Li, J.; Tinka, A.; Kiesel, S.; Durham, J. W.; Kumar, T. K. S.; and Koenig, S. 2021. Lifelong multi-agent path finding in large-scale warehouses. *Proceedings of the AAAI Conference on Artificial Intelligence* 35(13):11272–11281.

Liang, C.; Chee, K.; Zou, Y.; Zhu, H.; Causo, A.; Vidas, S.; Teng, T.; Chen, I.; Low, K.; and Cheah, C. 2015. Automated robot picking system for e-commerce fulfillment warehouse application. In *The 14th IFToMM World Congress*, volume 1.

Liu, M.; Ma, H.; Li, J.; and Koenig, S. 2019. Task and path planning for multi-agent pickup and delivery. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Ma, H., and Koenig, S. 2016. Optimal target assignment and path finding for teams of agents. *arXiv preprint arXiv:1612.05693*.

Ma, H.; Li, J.; Kumar, T.; and Koenig, S. 2017. Lifelong multi-agent path finding for online pickup and delivery tasks. *arXiv preprint arXiv:1705.10868*.

Pitanov, Y.; Skrynnik, A.; Andreychuk, A.; Yakovlev, K.; and Panov, A. 2023. Monte-carlo tree search for multi-agent pathfinding: Preliminary results. In *International Conference on Hybrid Artificial Intelligence Systems*, 649–660. Springer.

Silver, D.; Huang, A.; Maddison, C. J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. 2016. Mastering the game of go with deep neural networks and tree search. *nature* 529(7587):484–489.

Silver, D. 2021. Cooperative pathfinding. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 1(1):117–122.

Skrynnik, A.; Andreychuk, A.; Yakovlev, K.; and Panov, A. 2024. Decentralized monte carlo tree search for partially observable multi-agent pathfinding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 17531–17540.

Stern, R.; Sturtevant, N.; Felner, A.; Koenig, S.; Ma, H.; Walker, T.; Li, J.; Atzmon, D.; Cohen, L.; Kumar, T.; et al. 2019. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search*, volume 10, 151–158.

Tak, M. J. W.; Lanctot, M.; and Winands, M. H. M. 2014. Monte carlo tree search variants for simultaneous move games. In *2014 IEEE Conference on Computational Intelligence and Games*, 1–8.

Wurman, P. R.; D' Andrea, R.; and Mountz, M. 2008. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI Magazine* 29(1):9.