Multi-Robot Coordination and Layout Design for Automated Warehousing (Extended Abstract)*

Yulun Zhang[1], **Matthew C. Fontaine**[2], **Varun Bhatt**[2], **Stefanos Nikolaidis**[2], **Jiaoyang Li**[1]
[1]Robotics Institute, Carnegie Mellon University
[2]Thomas Lord Department of Computer Science, University of Southern California
yulunzhang@cmu.edu, {mfontain,vsbhatt,nikolaid}@usc.edu, jiaoyangli@cmu.edu

1 Introduction

Today, hundreds of robots are navigating autonomously in warehouses to transport goods from one location to another. Such warehouses are a multi-billion-dollar industry. To improve the throughput of automated warehouses, many works have studied the underlying lifelong Multi-Agent Path Finding (MAPF) problem for coordinating warehouse robots (Li et al. 2021). However, these works always use human-designed layouts, with an example shown in Figure 1a, to evaluate lifelong MAPF algorithms for automated warehouses. These human-designed layouts originated from the layouts for traditional warehouses where human workers, instead of robots, transport goods. They usually follow regularized patterns for human works to easily locate goods, but such patterns hardly matter for robots.

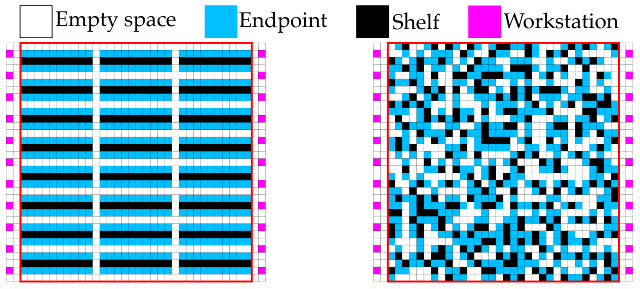
Therefore, instead of developing better lifelong MAPF algorithms, we propose to improve the throughput of automated warehouses by optimizing warehouse layouts, with an example shown in Figure 1b. We use the Quality Diversity (QD) algorithm MAP-Elites (Mouret and Clune 2015) to optimize the warehouse layouts with the objective of maximizing the throughput produced by a lifelong MAPF-based robot simulator while diversifying a set of user-defined measures, such as travel distances of the robots and the distribution of the shelves. In case the layout found is invalid, we follow previous work (Fontaine et al. 2021) and use a Mixed Integer Linear Programming (MILP) solver to repair it.

In this paper, we propose the first layout optimization method for automated warehouses based on MAP-Elites. We show that our optimized layouts greatly reduce the traffic congestion and improve the throughput compared to commonly used human-designed layouts.

2 Problem Definition

We first formally define the warehouse layouts. Following the terminology in the lifelong MAPF community, we use agents to refer to robots.

Definition 1 (Warehouse layout)**.** *We represent the warehouse layout as a four-neighbor grid, comprising four tile types: black tiles for shelves, blue tiles for endpoints next to shelves where agents interact with shelves, pink tiles for workstations where agents interact with human, and white tiles for empty spaces. Agents can traverse non-black tiles.*



(a) Human-designed (b) Optimized

Figure 1: Example of a commonly used human-designed layout and our optimized layout.

The task assigned to each agent is to move to blue and pink tiles alternatively to transport goods. To ensure the successful execution of the tasks, the layout must be *valid*.

Definition 2 (Valid layout)**.** *A warehouse layout is valid iff (1) any two tiles of blue or pink color are connected through a path with non-black tiles, (2) each blue tile is adjacent to at least one black tile, and (3) each black tile is adjacent to at least two blue tiles.*

Then, we define our optimization objective *throughput* and our layout optimization problem. For simplicity, we only optimize the *storage area*, namely the area inside the red boxes in Figure 1.

Definition 3 (Throughput)**.** *An agent finishes a task when it reaches its current assigned goal location. The throughput is the average number of finished tasks per timestep.*

Definition 4 (Layout Optimization)**.** *Given the layout of a non-storage area and a desired number of shelves N_s, the layout optimization problem searches for the allocation of tiles inside the storage area to find valid layouts with N_s black tiles while maximizing their throughput and diversifying their user-defined diversity measures.*

*The full paper can be found in the IJCAI 2023 proceeding at https://doi.org/10.24963/ijcai.2023/611.

Setup	S_{sa}	$\|\mathbf{X}\|$	S_f	N_s	N_a	P_a
1	12×17	3^{204}	16×17	40	90	39%
2	32×33	3^{1056}	36×33	240	200	20%

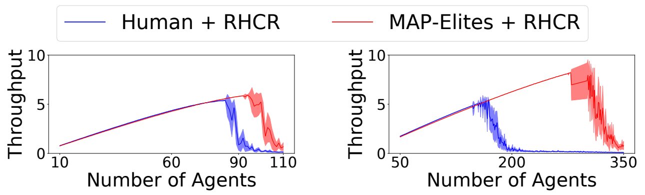
Table 1: Summary of the experiment setup. S_{sa} is the size of the storage area. $|\mathbf{X}|$ is the number of all possible layouts without constraints, S_f is the full size of the layout, N_s is the number of shelves, N_a is the number of agents, and $P_a = \frac{N_a}{S_f - N_s}$ is the agent density, computed as the percentage of traversable tiles (i.e., non-black tiles) occupied by agents.

3 Layout Optimization Approach

QD Formulation QD algorithms simultaneously optimize an objective function while diversifying a set of diversity measure functions. We search over the possible tile combinations. We represent the storage area of a warehouse layout as a vector of discrete variables $\vec{x} \in \mathbf{X}$, where each discrete variable $x_i \in \{black, blue, white\}$ corresponds to the tile type of the i^{th} tile in the layout, and $\mathbf{X}$ is the space of all possible layouts. Our objective function runs a lifelong MAPF simulator on the input layout for N_e times and returns the average throughput. Our diversity measure functions compute (1) the number of connected shelf components and (2) the average length of tasks in the layout.

MAP-Elites MAP-Elites (Mouret and Clune 2015) is a popular QD algorithm. It discretizes the measure space defined by the measure functions, referred to as an *archive*, and searches for the best solution in each discretized cell, referred to as an *elite*. We use MAP-Elites to search for the elite layouts by iteratively updating the archive. In each iteration, we choose a batch of b elite layouts from the archive uniformly with replacement. Inspired by a previous work (Fontaine et al. 2021), we mutate each of the b elite layouts by uniformly selecting k tiles from each layout and changing the tile type of each of them to a random tile type. The variable k is sampled from a geometric distribution with $P(X = k) = (1-p)^{k-1}p$ with $p = \frac{1}{2}$. When the archive is empty in the first iteration, we generate b random layouts. Then, we repair the layouts with a MILP solver to ensure that they are valid (see details below). We then extract the average throughput and measures by running the lifelong MAPF simulator N_e times, each runs for T timesteps with N_a agents. Finally, we add the evaluated layouts to the corresponding cells in the archive if their throughput is larger than that of the elite layouts in the same cells. We run MAP-Elites for I iterations with batch size b, resulting in a total of $N_{eval} = b \times I$ evaluations.

MILP Repair After we generate or mutate a layout, we add the non-storage area back to the layout and repair it so that it becomes valid. We follow the previous work (Fontaine et al. 2021) and formulate the repair as a MILP. We minimize the hamming distance between the unrepaired layout $\vec{x}_{in}$ and the repaired layout $\vec{x}_{out}$ while asking $\vec{x}_{out}$ to satisfy the following constraints: (1) the non-storage area of $\vec{x}_{out}$ is kept unchanged, (2) $\vec{x}_{out}$ is valid, and (3) the number of shelves in $\vec{x}_{out}$ is equal to N_s.



(a) Setup 1 (b) Setup 2

Figure 2: Throughput with different numbers of agents. The solid line shows average and the shaded area is the 95% confidence interval.

4 Experimental Evaluation

Experiment Setup Table 1 summarizes the setup. We use two map sizes to show how our methods scale with the size of the maps. We use RHCR (Li et al. 2021), a state-of-the-art centralized lifelong MAPF algorithm to evaluate the throughput. RHCR replans paths for all agents at every h timesteps, and the planned paths are provably collision-free for w timesteps. We use $w = 10$ and $h = 5$ with PBS (Ma et al. 2019) as the MAPF solver. The goal locations alternate between randomly chosen workstations and endpoints. For MAP-Elites, we set $b = 50$, $N_{eval} = 10,000$, $T = 1,000$, and $N_e = 5$. We stop the lifelong MAPF simulation early if (unrecoverable) *congestion* occurs, which happens when more than half of the agents take wait actions.

Results Figure 2 shows the throughput with different numbers of agents on layouts that are optimized with N_a agents. In comparison to the human-designed layouts, our optimized layouts achieve similar throughput for small numbers of agents and significantly better throughput for large numbers of agents. We double the scalability of RHCR from fewer than 150 agents to more than 300 agents in setup 2. However, the improvement is less significant in setup 1 due to larger agent density (P_a).

Acknowledgements

This work used Bridge-2 at Pittsburgh Supercomputing Center (PSC) through allocation CIS220115 from the Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation grants #2138259, #2138286, #2138307, #2137603, and #2138296.

References

Fontaine, M. C.; Hsu, Y.-C.; Zhang, Y.; Tjanaka, B.; and Nikolaidis, S. 2021. On the Importance of Environments in Human-Robot Coordination. In *RSS*.

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. In *AAAI*, 11272–11281.

Ma, H.; Harabor, D.; Stuckey, P. J.; Li, J.; and Koenig, S. 2019. Searching with Consistent Prioritization for Multi-Agent Path Finding. In *AAAI*, 7643–7650.

Mouret, J.-B.; and Clune, J. 2015. Illuminating search spaces by mapping elites. *ArXiv*, abs/1504.04909.