

# Multi-Robot Coordination and Layout Design for Automated Warehousing\*

Yulun Zhang<sup>1</sup>, Matthew C. Fontaine<sup>2</sup>, Varun Bhatt<sup>2</sup>, Stefanos Nikolaidis<sup>2</sup> and Jiaoyang Li<sup>1</sup>

<sup>1</sup>Robotics Institute, Carnegie Mellon University

<sup>2</sup>Department of Computer Science, University of Southern California

yulunzhang@cmu.edu, {mfontain,vsbhatt,nikolaid}@usc.edu, jiaoyangli@cmu.edu

## Abstract

With the rapid progress in Multi-Agent Path Finding (MAPF), researchers have studied how MAPF algorithms can be deployed to coordinate hundreds of robots in large automated warehouses. While most works try to improve the throughput of such warehouses by developing better MAPF algorithms, we focus on improving the throughput by optimizing the warehouse layout. We show that, even with state-of-the-art MAPF algorithms, commonly used human-designed layouts can lead to congestion for warehouses with large numbers of robots and thus have limited scalability. We extend existing automatic scenario generation methods to optimize warehouse layouts. Results show that our optimized warehouse layouts (1) reduce traffic congestion and thus improve throughput, (2) improve the scalability of the automated warehouses by doubling the number of robots in some cases, and (3) are capable of generating layouts with user-specified diversity measures.<sup>1</sup>

## 1 Introduction

Today, hundreds of robots are navigating autonomously in warehouses to transport goods from one location to another [Wurman *et al.*, 2007; Kou *et al.*, 2020]. Such automated warehouses are currently a multi-billion-dollar industry led by big companies like Amazon, Alibaba, and Ocado. To improve the throughput of automated warehouses, a lot of works have studied the underlying Multi-Agent Path Finding (MAPF) problem for coordinating warehouse robots smoothly [Nguyen *et al.*, 2017; Ma *et al.*, 2017; Liu *et al.*, 2019; Li *et al.*, 2020; Kou *et al.*, 2020; Chen *et al.*, 2021; Contini and Farinelli, 2021; Varambally *et al.*, 2022; Damani *et al.*, 2021].

These works always use human-designed layouts, with an example shown in Figure 1a and in Appendix A, to evaluate MAPF algorithms for automated warehouses. These human-designed layouts originated from the layouts for traditional

\*The full version of the paper can be found in <https://arxiv.org/abs/2305.06436>

<sup>1</sup>We include the source code of the experiments in: [https://github.com/lunjohnzhang/warehouse\\_env\\_gen\\_public](https://github.com/lunjohnzhang/warehouse_env_gen_public)

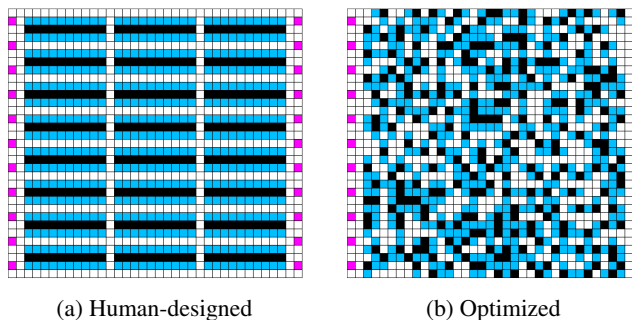


Figure 1: Example of layout optimization. (a) and (b) illustrate a commonly used human-designed layout and our optimized layout, respectively. Robots move to blue and pink tiles alternatively to transport goods. Black tiles are shelves, which the robots cannot traverse.

warehouses where human workers, instead of robots, transport goods. They usually follow regularized patterns with shelves clustered in rectangular shapes. Such patterns make it easy for human workers to locate goods. But, for robots, it hardly matters to have such regularized patterns. We thus abandon pattern regularity to explore novel layouts with better throughput. Figure 1b shows one of our optimized layouts with the same number of shelves.

Therefore, instead of developing better MAPF algorithms, we propose to improve the throughput of automated warehouses by optimizing warehouse layouts. We use the Quality Diversity algorithm MAP-Elites [Mouret and Clune, 2015] to optimize the warehouse layouts with the objective of maximizing the throughput produced by a 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 by MAP-Elites is invalid, we follow previous work [Zhang *et al.*, 2020; Fontaine *et al.*, 2021a] and use a Mixed Integer Linear Programming (MILP) solver to repair it. Since both the robot simulator and the MILP solver are time-consuming, we combine them with an online trained surrogate model using Deep Surrogate Assisted Generation of Environments (DSAGE) [Bhatt *et al.*, 2022].

We make the following contributions: (1) we show that, even with state-of-the-art MAPF algorithms, commonly used

human-designed layouts can lead to congestion for warehouses with large numbers of robots, (2) we propose the first layout optimization method for automated warehouses based on MAP-Elites and DSAGE, (3) we extend DSAGE by scaling it from single-agent domains to the warehouse domain with up to 200 robots and incorporating MILP to ensure that the layouts satisfy warehouse-specific constraints, (4) we show that our optimized layouts greatly reduce the traffic congestion and improve the throughput compared to commonly used human-designed layouts, and (5) we show that our proposed layout optimization method can generate a diverse set of high throughput layouts with respect to user-defined diversity measures.

## 2 Problem Definition

We formally define the warehouse layouts as follows with two examples shown in Figure 2. Following the terminology in the MAPF community, we use agents to refer to robots.

**Definition 1** (Warehouse layout). *We represent the warehouse layout with a four-neighbor grid, where each tile can be one of the following five types. Black tiles represent shelves to store packages. Blue tiles represent endpoints where agents park and interact with the adjacent shelves. Pink tiles represent workstations where agents interact with human workers. White tiles represent empty spaces. Orange tiles represent home locations for agents, which are empty spaces with special properties used by some MAPF algorithms. Agents can traverse non-black tiles.*

Since the orange, blue, and pink tiles are the potential start and goal locations of the agents, a valid warehouse layout needs to make these tiles connected to each other. Furthermore, since the agents interact with the shelves from the endpoints, a valid warehouse layout needs to also make blue and black tiles next to each other.

**Definition 2** (Valid layout). *A warehouse layout is valid iff (1) any two tiles of blue, pink, or orange 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.*

We further define the well-formed property [Liu *et al.*, 2019] for warehouse layouts, a required property by some MAPF algorithms.

**Definition 3** (Well-formed layout). *A valid warehouse layout is well-formed iff the number of orange tiles is no smaller than the number of agents, and any two tiles of blue or orange color are connected through a path with only white tiles.*

We pick two common warehouse scenarios to test our layout optimization methods, one requiring well-formed layouts and one requiring valid layouts.

**Home-Location Scenario** Following previous work [Liu *et al.*, 2019; Li *et al.*, 2021], the home-location scenario simulates a warehouse in which the agents transport goods from one shelf to another. That is, the agents are asked to move from one blue tile to another. Figure 2a shows an example. We require the layouts to be well-formed.

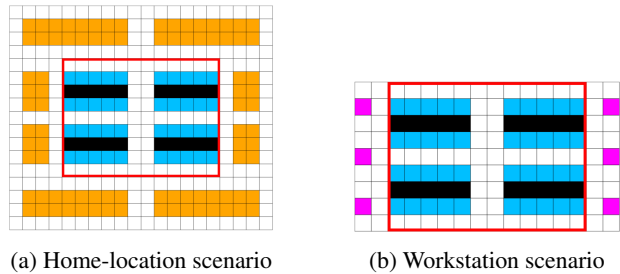


Figure 2: Example layouts. We optimize tiles inside the red box.

**Workstation Scenario** Following previous work [Hönig *et al.*, 2019], the workstation scenario simulates a warehouse in which the agents transport goods between workstations and shelves. That is, the agents are asked to move to pink and blue tiles alternately. Figure 2b shows an example. We require the layouts to be valid.

Finally, we define our optimization objective *throughput* and our layout optimization problem. Our goal is to find novel distributions of the shelves and their associated endpoints. We thus keep the *non-storage area*, i.e., area outside of the red box in Figure 2, unchanged for simplicity and only optimize the *storage area*, i.e., area inside the red box.

**Definition 4** (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 5** (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 black, blue, and white tiles inside the storage area to find valid/well-formed layouts with  $N_s$  black tiles while maximizing their throughput and diversifying their user-defined diversity measures.*

## 3 Preliminaries

We now introduce the (lifelong) MAPF algorithms that we use in our simulator and existing works on warehouse layout optimization and scenario generation with Quality Diversity algorithms.

### 3.1 Lifelong Multi-Agent Path Finding

Multi-Agent Path Finding (MAPF) [Stern *et al.*, 2019] aims to find collision-free paths for a group of agents from their given start locations to goal locations. Each agent has only one pair of start and goal locations and, after it reaches its goal location, waits there forever. Lifelong MAPF [Li *et al.*, 2021] is a variant of MAPF in which the agents do not stop and wait at their goal locations. Instead, a new goal location is assigned once an agent reaches its current goal location so that the agents keep moving constantly. In this work, we use a widely-used lifelong MAPF definition [Li *et al.*, 2021] where (1) the agents move on a four-neighbor grid, (2) in each discrete timestep, every agent can either stay at its current tile or move to an adjacent tile, (3) the agents cannot be at the same tile or swap their locations at adjacent tiles at the same timestep, and (4) the goal is to maximize the throughput.

Previous works have approached the lifelong MAPF problem by solving it as a whole [Nguyen *et al.*, 2017] or decomposing it into a sequence of MAPF instances and solving them on the fly [Cáp *et al.*, 2015; Ma *et al.*, 2017; Wan *et al.*, 2018; Liu *et al.*, 2019; Grenouilleau *et al.*, 2019; Li *et al.*, 2021]. We choose two state-of-the-art lifelong MAPF algorithms of two types as representative algorithms in our experiments, namely Dummy-Path Planning (DPP) [Liu *et al.*, 2019] and Rolling-Horizon Collision Resolution (RHCR) [Li *et al.*, 2021].

**DPP** DPP plans paths for idle agents (i.e., agents that reach their current goal locations and need paths to their new goal locations) at every timestep. To ensure that there always exist paths for idle agents that do not collide with the paths of non-idle agents, it requires the layout to be well-formed and thus can be used only in the home-location scenario.

**RHCR** RHCR replans paths for all agents at every  $h$  timesteps, and the planned paths are provably collision-free for  $w$  timesteps. It does not require the layout to be well-formed and thus can be used in both warehouse scenarios.

### 3.2 Existing Work on Layout Optimization

To the best of our knowledge, no previous works have studied the warehouse layout optimization problem as described in Definition 5 and explored the possibilities of having non-regularized warehouse layouts. Some related works include using queuing network models to estimate the throughput of a given warehouse layout [Wu *et al.*, 2020; Lamballais *et al.*, 2017] and guide the design of warehouse layouts. A recent work [Yang *et al.*, 2021] proposed using integer programming to optimize the locations of the workstations. However, it keeps the shelves fixed and evaluates the quality of the layouts by the average distance between the workstations and shelves, which does not reflect traffic congestion and is thus less accurate than our optimization objective throughput.

### 3.3 Automatic Environment Generation and Quality-Diversity (QD) Optimization

Automatic environment generation algorithms have been proposed in different fields such as procedural content generation [Shaker *et al.*, 2016], reinforcement learning [Risi and Togelius, 2019; Justesen *et al.*, 2018], and robotics [Fontaine and Nikolaidis, 2021]. Among the environment generation techniques, QD algorithms [Fontaine *et al.*, 2021a; Fontaine and Nikolaidis, 2021; Fontaine *et al.*, 2021b] are popular because they can find a diverse set of high-quality solutions with user-specified objectives and diversity measures.

MAP-Elites [Cully *et al.*, 2015; Mouret and Clune, 2015] is a popular QD algorithm. It discretizes the measure space, which is defined by the user-specified diversity measure functions, and searches for the best solution in each discretized cell. The resulting best solutions in each cell become a set of diverse and high-quality solutions. Deep Surrogate Assisted MAP-Elites (DSA-ME) [Zhang *et al.*, 2022] improves MAP-Elites by using a surrogate model based on deep neural networks to speed up the solution evaluation process. DSAGE [Bhatt *et al.*, 2022] then extends DSA-ME to the environment generation problem for single-agent domains.

## 4 Layout Optimization Approach

We first formulate the layout optimization problem as a QD problem and introduce our basic framework based on MAP-Elites, which searches over the tiles of the warehouse layout and uses a MILP solver to make the layout valid or well-formed. We then introduce our advanced framework based on DSAGE, which further introduces a surrogate model to improve optimization efficiency.

### 4.1 MAP-Elites

Since our warehouse layout is made of discrete tiles, we directly search over the possible tile combinations through a QD algorithm. Specifically, 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  $i^{\text{th}}$  tile in the layout, and  $\mathbf{X}$  is the space of all possible layouts.

To formulate layout optimization as a QD problem, we define an objective function to be maximized,  $f : \mathbf{X} \rightarrow \mathbb{R}$ , that runs a lifelong MAPF simulator on the input layout for  $N_e$  times and returns the average throughput. We also define a measure function  $\mathbf{m} : \mathbf{X} \rightarrow \mathbb{R}^m$  that quantifies characteristics of the layout to diversify. In our experiment, we select the number of connected shelf components and the average length of tasks in the layout as the diversity measures. We chose connected self components because shelves are often organized into “islands” of shelves in human-designed layouts. We formalize connected shelf components as a set of shelves such that any pair of shelves in the set can be connected by a path through only the shelves in the set. In addition, we pick average task length because it is one of the factors that could affect throughput. We define the average length of tasks as the average distance between any pair of blue and pink tiles for the workstation scenario and any pair of two blue tiles for the home-location scenario.

We discretize the measure space evenly into  $M$  cells and aim to find, for each cell  $j$ , the best layout  $\vec{x}_j$ , called an *elite layout*, that maximizes  $f(\vec{x}_j)$ . We call this discretized measure space with elites an *archive*. The bottom plot in Figure 7 shows an example. Solving this problem generates a set of layouts with high throughput and different numbers of connected shelf components and average lengths of tasks.

**Main Algorithm** We maintain an archive and improve its elites through iterations. In each iteration, we choose a batch of  $b$  elite layouts from the archive uniformly with replacement, where the batch size  $b$  is a hyperparameter. Inspired by previous work [Fontaine *et al.*, 2019; Fontaine *et al.*, 2021a], 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  $p = \frac{1}{2}$ . For the first iteration when the archive is empty, we just generate  $b$  random layouts. Then, we repair the layouts with a MILP solver to ensure that they are valid or well-formed (see details below). After getting the repaired layouts, we 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 lay-

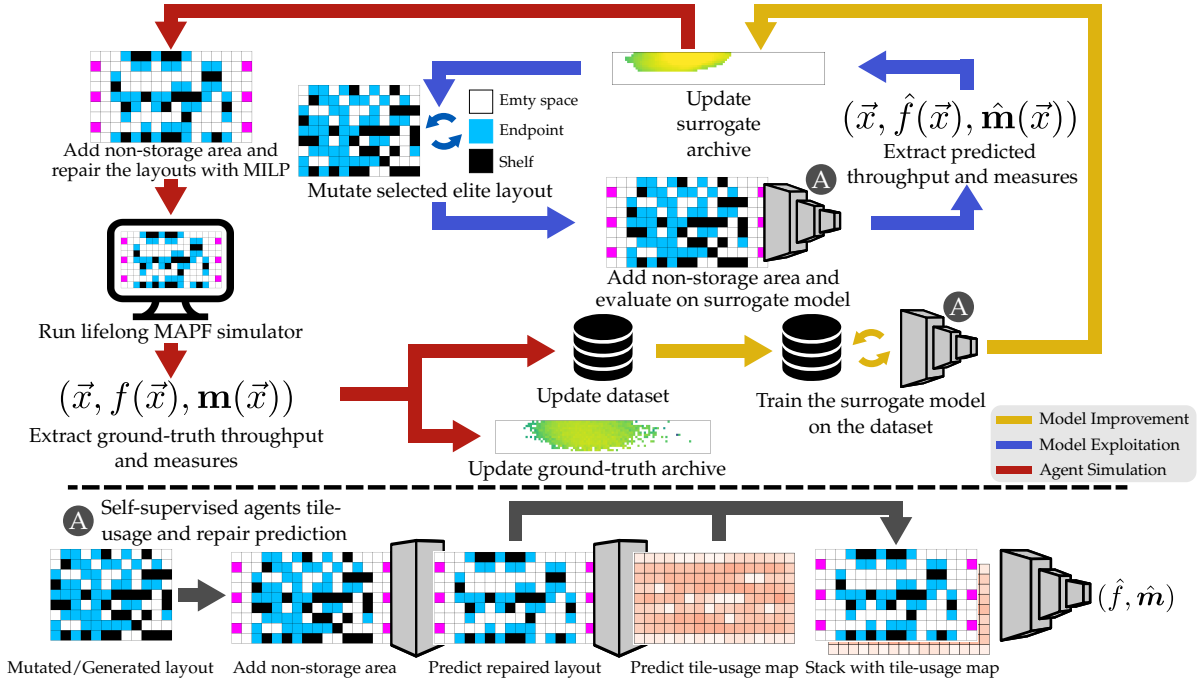


Figure 3: Overview of the warehouse layout optimization with extended DSAGE. We start by generating a randomly generated set of layouts and training the initial surrogate model. We then iterate over the model exploitation, agent simulation, and model improvement phases until the number of evaluations in the lifelong MAPF simulator reaches  $N_{eval}$ .

outs 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, where each evaluation consists of  $N_e$  simulations on the lifelong MAPF simulator. Figure 8 in Appendix B gives an overview of MAP-Elites.

**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 or well-formed. We follow previous work [Zhang *et al.*, 2020; Fontaine *et al.*, 2021a] and formulate the repair process as a Mixed Integer Linear Programming (MILP) problem. In particular, we minimize the hamming distance between the input unrepaired layout  $\vec{x}_{in}$  and the output repaired layout  $\vec{x}_{out}$  while asking  $\vec{x}_{out}$  to satisfy the following constraints: (1) the tiles of  $\vec{x}_{out}$  in the non-storage area are kept unchanged, (2)  $\vec{x}_{out}$  is valid for the workstation scenario and well-formed for the home-location scenario, and (3) the number of shelves in  $\vec{x}_{out}$  is equal to a predefined number  $N_s$ , which ensures that  $\vec{x}_{out}$  has the desired storage capability. We constrain the number of shelves in the MILP repair step instead of in MAP-Elites so that the QD algorithm can search freely the solution space without being constrained to  $N_s$ . The detailed formulation is shown in Appendix C.

## 4.2 DSAGE

Both the MILP solver and the lifelong MAPF simulator can be time-consuming for large layouts with large numbers of agents. For instance, in our experiments, the MILP solver can take as long as two minutes to repair one layout, and

the lifelong MAPF simulator can take up to ten minutes to evaluate one layout. Therefore, we extend Deep Surrogate Assisted Generation of Environment (DSAGE) [Bhatt *et al.*, 2022] as an additional method to optimize warehouse layouts, which exploits an online trained deep surrogate model to guide MAP-Elites and improve its sample efficiency. To use DSAGE for layout optimization, we make the following extensions to DSAGE: (1) our model predicts the tile usage of hundreds of agents instead of a single agent, (2) the tile usage depends on the repaired layout, thus our surrogate model includes an additional component that predicts the output of the MILP solver.

Figure 3 gives an overview of our extended DSAGE framework, which consists of three phases (model exploitation, agent simulation, and model improvement), and a surrogate model. We start by generating a set of  $n_{rand}$  random layouts and using them to train a surrogate model, which predicts the outputs of the MILP solver and the lifelong MAPF simulator for given unrepaired layouts. The model exploitation phase runs MAP-Elites with the surrogate model, instead of the MILP solver and the MAPF simulator, and gets a surrogate archive of (predicted) elite layouts. For every  $n$  iterations of the model exploitation phase, we run the agent simulation phase, which repairs and evaluates the elite layouts from the surrogate archive using the MILP solver and the MAPF simulator and adds them to a ground-truth archive. The model improvement phase then saves all evaluated layouts in a dataset and uses it to improve the surrogate model. We run this process iteratively until  $N_{eval}$  layouts are evaluated on the lifelong MAPF simulator. This way, we simultaneously generate

a diverse set of high-quality layouts and train a deep surrogate model to predict the throughput and measures of given layouts. We show the details of the three phases below.

**Model Exploitation** At this phase, we run MAP-Elites by replacing the MILP solver and the lifelong MAPF simulator with the surrogate model. That is, instead of computing the *ground-truth* throughput  $f(\vec{x})$  and measures  $\mathbf{m}(\vec{x})$  using a repaired valid layout, we query the surrogate model for the *predicted* throughput  $\hat{f}(\vec{x})$  and measures  $\hat{\mathbf{m}}(\vec{x})$  using an unrepaired layout. Then the model exploitation phase efficiently builds a *surrogate* archive, which contains the layouts that the surrogate model predicts to be high-performing and diverse.

**Agent Simulation** At this phase, we repair and evaluate the elite layouts from the surrogate archive to update the ground-truth archive. Since the surrogate archive sometimes contains a large number of elite layouts, we use the downsampling method from [Bhatt *et al.*, 2022] to select a subset of elite layouts to be repaired and evaluated. Specifically, we uniformly divide the archive into sub-areas and uniformly select a layout in each sub-area. Besides updating the ground-truth archive, we also store the evaluated layouts in the dataset.

**Model Improvement** At this phase, we fine-tune the surrogate model with the data in the dataset. We extend the surrogate model from previous work [Bhatt *et al.*, 2022] and use a three-stage deep convolutional neural network with a self-supervision procedure as our surrogate model. The architecture of the model is shown at the bottom of Figure 3. It contains three sub-networks  $s_1$ ,  $s_2$ , and  $s_3$ . We first use  $s_1$  to predict a repaired layout given an input unrepaired layout. We then use  $s_2$  to predict a tile-usage map of the predicted repaired layout. A tile-usage map records the number of timesteps each tile is occupied by an agent in the lifelong MAPF simulation. We then normalize the tile-usage map so that the numbers sum up to 1, forming a distribution of the tile usage in the layout. We use the tile-usage map to self-supervise the training of the surrogate model because we observe that the distribution of the tile usage is correlated to the throughput of the layout. For example, Figure 4 shows the tile-usage maps of the two layouts in Figures 1a and 1b with 200 agents. Figure 4a indicates that the agents are congested on the right part of the warehouse, while Figure 4b has no sign of congestion. After predicting the tile-usage map, we concatenate the predicted tile-usage map with the predicted repaired layout and use  $s_3$  to predict the throughput and measures. We use a convolutional neural network with 48 layers and 974,403 parameters as the surrogate model, with its detailed architecture and training summarized in Appendix D.1.

## 5 Experimental Evaluation

In this section, we evaluate MAP-Elites and DSAGE and compare the optimized layouts with the human-designed ones.

### 5.1 Experiment Setup

Table 1 summarizes the experiment setup. Columns 2-6 show the configurations related to the layouts. We use one map size for the home-location scenario and three map sizes for the

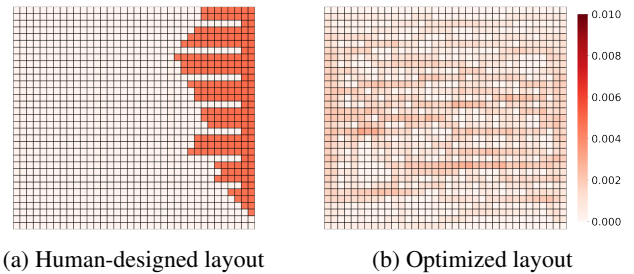


Figure 4: Tile-usage maps of the two layouts in Figures 1a and 1b.

Setup	$S_{sa}$	$ \mathbf{X} $	$S_f$	$N_s$	Scenario	MAPF RHCR DPP	$N_a$	$P_a$
1	$12 \times 9$	$3^{108}$	$20 \times 17$	20	Home location		88	28%
2	$12 \times 9$ (small)	$3^{108}$	$16 \times 9$	20	Workstation	RHCR	60	48%
3	$12 \times 17$ (medium)	$3^{204}$	$16 \times 17$	40			90	39%
4	$32 \times 33$ (large)	$3^{1056}$	$36 \times 33$	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 percentage of traversable tiles (i.e., non-black tiles) that are occupied by agents.

workstation scenario (to show how our methods scale with the size of the maps). Notably, setup 4 has the same storage area size as a commonly used human-designed layout in previous works [Li *et al.*, 2021]. The number of shelves  $N_s$  shown in column 5 is chosen to be the same as those in the human-designed layouts so that the optimized layouts have the same storage capacity as the human-designed layouts.

Column 7 shows the lifelong MAPF simulator with RHCR and DPP. Following design recommendation of previous work [Li *et al.*, 2021], we use PBS [Ma *et al.*, 2019] as the MAPF solver with  $w = 10$  and  $h = 5$  in RHCR, and use prioritized planning [Erdmann and Lozano-Perez, 1986] as the MAPF solver in DPP. Both of them use SIPP [Phillips and Likhachev, 2011] as the single-agent solver. In the workstation scenario, the initial locations of the agents are uniformly chosen at random from non-shelf tiles, and the task assigner alternates every agent’s goal location between a workstation and an endpoint, both chosen uniformly at random. In the home-location scenario, the initial locations of the agents are uniformly chosen at random from the home locations, and the task assigner assigns uniformly random goal locations from the endpoints. Since DPP can only work on well-formed layouts, we use it only for the home-location scenario.

Columns 8 and 9 show the number of agents  $N_a$  and the agent density  $P_a$ . For the home-location scenario, we choose  $N_a = 88$  because we have only 88 home locations. For the workstation scenario, we run the MAPF simulator on the human-designed layouts with an increasing number of agents and choose  $N_a$  when we observe a significant drop in throughput.

For both MAP-Elites and DSAGE, we set  $b = 50$ ,  $N_{eval} = 10,000$ ,  $T = 1,000$ , and  $N_e = 5$ . For DSAGE, we further set  $n_{rand} = 500$  and  $n = 10,000$  for setup 1 and 2 and  $n = 50,000$  for setup 3 and 4. We stop the lifelong MAPF simulation early (i.e., before timestep  $T$ ) if (unrecov-

Setup	QD + MAPF Algo	SR	Throughput	CPU Runtime
1	<b>DSAGE + RHCR</b>	<b>100%</b>	<b>6.34 ± 0.01</b>	174.62 ± 1.97
	MAP-Elites + RHCR	<b>100%</b>	5.33 ± 0.00	382.80 ± 2.75
	Human + RHCR	0%	N/A	N/A
	DSAGE + DPP	<b>100%</b>	5.06 ± 0.00	<b>46.42 ± 0.99</b>
	MAP-Elites + DPP	<b>100%</b>	4.79 ± 0.00	47.14 ± 1.41
2	Human + DPP	<b>100%</b>	4.36 ± 0.00	55.71 ± 0.96
	<b>DSAGE + RHCR</b>	30%	<b>4.28 ± 0.02</b>	<b>73.13 ± 1.61</b>
	MAP-Elites + RHCR	<b>80%</b>	4.08 ± 0.02	83.36 ± 1.36
3	Human + RHCR	0%	N/A	N/A
	<b>DSAGE + RHCR</b>	<b>100%</b>	<b>5.91 ± 0.00</b>	<b>93.66 ± 0.42</b>
	MAP-Elites + RHCR	<b>100%</b>	5.80 ± 0.00	103.55 ± 0.47
4	Human + RHCR	0%	N/A	N/A
	<b>DSAGE + RHCR</b>	<b>100%</b>	<b>6.30 ± 0.00</b>	<b>131.63 ± 3.32</b>
	MAP-Elites + RHCR	<b>100%</b>	6.27 ± 0.00	152.13 ± 1.16

Table 2: Success rates, throughput, and CPU runtimes. *SR* refers to the success rate, which is the percentage of simulations that end without congestion. *CPU runtime* refers to the runtime, in seconds, of the 5000-timestep simulation. We only measure the throughput and CPU runtime of successful simulations.

erable) *congestion* occurs, which is defined as the case where more than half of the agents take wait actions at a particular timestep. This is because (1) we penalize the layouts that run into congestion and (2) we empirically observe that congestion can quickly decrease the number of finished tasks, and keeping running the simulation with congestion seldom increases the throughput. We summarize other hyperparameters and compute resources in Appendix D.2.

To report robust results, we run the lifelong MAPF simulator on every human-designed or optimized layout that we evaluate in this section with  $T_{eval} = 5,000$  timesteps for, unless explicitly stated otherwise, 10 times, using machines specified in Appendix D.3. We report both average results and their distributions (in the format of  $x \pm y$  for numerical results with standard deviations and solid lines with shaded areas for graphical results with 95% confidence intervals).

## 5.2 Results

### Comparing Optimized and Human-Designed Layouts

We take the best layout from the archive for each setup and compare them in Table 2. The results show that our optimized layouts have significantly better throughput and less congestion than the human-designed layouts in all cases. For example, we scale RHCR from complete failure to complete success for all setups except for setup 2. Moreover, we find that, for the same lifelong MAPF algorithm, a layout with better throughput often leads to a smaller CPU runtime of the lifelong MAPF simulator. Thus, we also reduce the runtime of DPP in the setup 1.

To further understand the gap of the success rates and throughput among different layouts, we plot the simulation process in Figures 5a to 5d. Here, we do not stop the simulation even if they run into congestion and report results over 100 runs (instead of 10 runs). As shown in the figures, the human-design layouts often encounter severe unrecoverable congestion, leading to a significant drop in the number of finished tasks over time. In contrast, our optimized layouts are stable in most cases. The only exception is setup 2, where the performance of our optimized layouts also decreases over

time, though with a smaller decreasing speed than that of the human-design layout. This is because setup 2 has the highest agent density (see  $P_a$  in Table 1), which makes it hard to find layouts that do not cause congestion.

### Generalization and Scalability

Although we run MAP-Elites and DSAGE with a fixed number of agents  $N_a$ , the optimized layouts can also be applied to other numbers of agents. Figures 5e to 5h 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 in setup 1. As for the workstation scenario (i.e., setups 2-4), the trend is similar, and the larger the layout, the bigger the improvement. For instance, we double the scalability of RHCR from fewer than 150 agents to more than 300 agents in setup 4.

One reason why we do not improve the throughput for small numbers of agents is that our layouts are optimized using large numbers of agents. To justify this, we optimize our layouts using three different numbers of agents in setup 1 and report the results in Figure 6. As shown, when we use a small number of agents to optimize the layout, the resulting layout leads to better throughput for small numbers of agents.

We also observe an interesting trade-off between RHCR and DPP. Previous work [Li *et al.*, 2021] shows that RHCR leads to better throughput than DPP, which matches our results for small numbers of agents. But for large numbers of agents, we observe that RHCR is more prone to congestion and thus performs worse than DPP. Nevertheless, our optimized layouts improve the performance of both of them.

### Diversity in Generated Layouts

A major benefit of using QD algorithms to optimize warehouse layouts is to get a diverse set of high-throughput layouts with user-defined diversity measures. Figure 7 shows two example optimized layouts with different numbers of connected shelf components. The layout on the left clustered more shelves together, forming 5 components, whereas the layout on the right scatters the shelves in the storage area, forming 19 components. Notably, both layouts achieve higher throughput than the human-designed layout. In addition, the optimized layouts allocate the shelves and endpoints on the borders of the storage area. The geometric structure allows the MAPF algorithms to improve throughput.

We observe that DSAGE is more capable of generating diverse layouts in the archive than MAP-Elites. For example, in setup 4, although there is no significant difference between the throughput of the best optimized layouts generated by MAP-Elites and DSAGE with 200 agents, DSAGE generates 1,317 elite layouts in the archive, and MAP-Elites generates only 1,039. As a result, DSAGE generates layouts with numbers of connected shelf components ranging from 146 to 218, while MAP-Elites cannot generate layouts with fewer than 153 or more than 207 connected shelf components. We show more results on DSAGE and MAP-Elites in Appendix E.

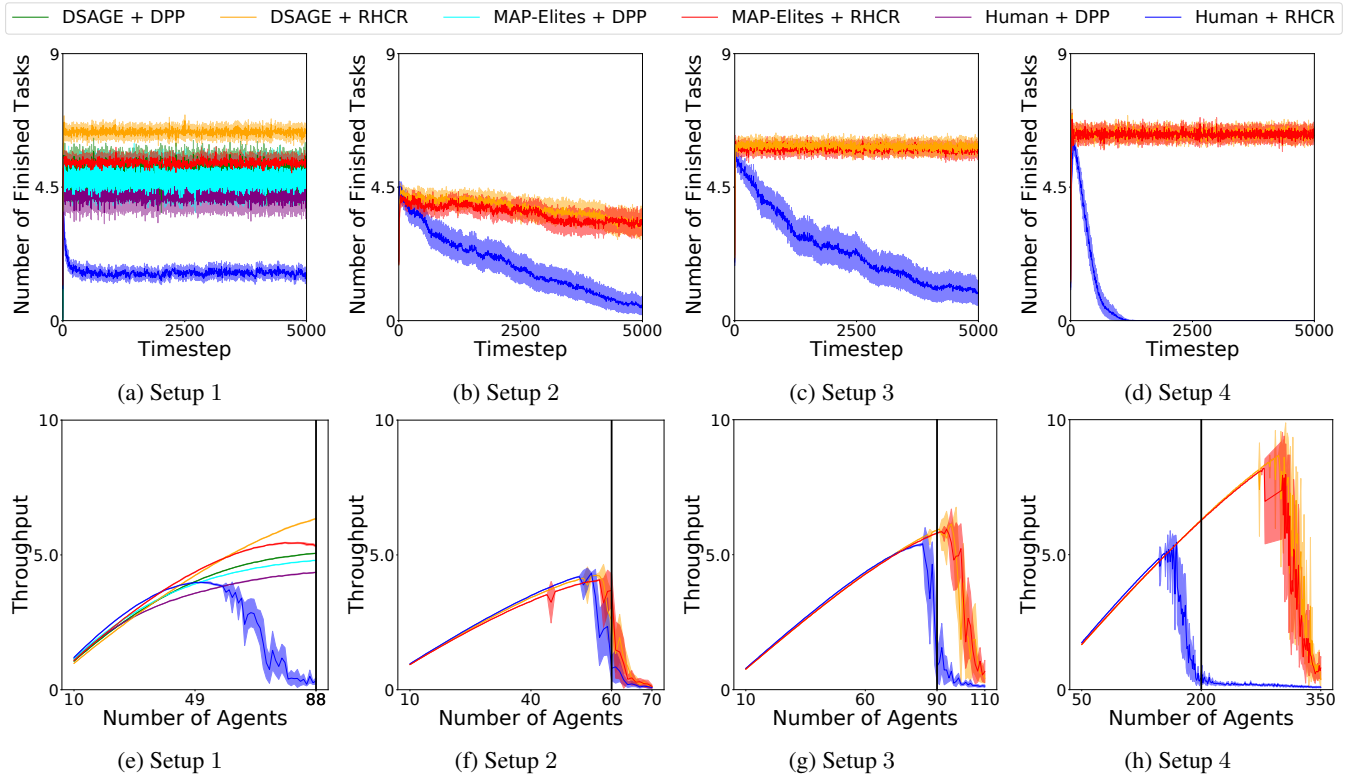


Figure 5: Number of finished tasks per timestep with  $N_a$  agents and throughput with different numbers of agents. The layouts are optimized with  $N_a$  agents, where  $N_a$  is indicated by the vertical lines in Figures 5e to 5h.

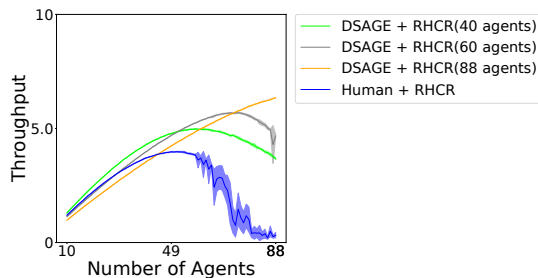


Figure 6: Throughput for layouts optimized with different numbers of agents for setup 1.

## 6 Conclusion

We present the first layout optimization approach for automated warehouses that explores novel layouts with non-regularized patterns and explicitly optimizes throughput. We extend DSAGE from single-agent domains to the warehouse domain with as many as 200 agents. We also incorporate MILP to DSAGE and extend the surrogate model to predict repaired layouts. Our result shows that the optimized layouts outperform commonly used human-designed layouts in throughput and CPU runtime. Our methods work with different warehouse scenarios, different MAPF algorithms, and layouts of different sizes. In addition, our methods provide diverse layouts with respect to user-specified diversity measures.

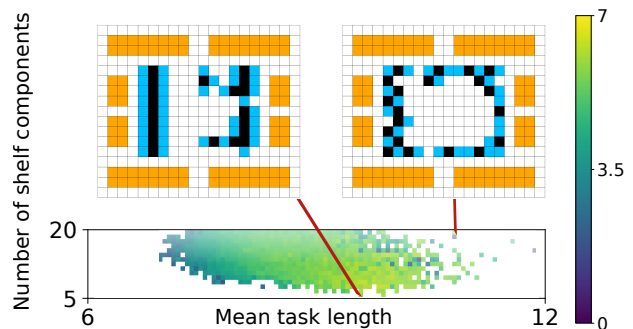


Figure 7: Example optimized layouts with different numbers of connected shelf components from the archive. Layouts are optimized with DSAGE and RHCR with  $N_a = 88$  agents. The layouts on the left and right achieved throughput of 6.03 and 4.80, respectively.

Our work has many future directions. First, we can incorporate the optimization of the non-storage area into our method because previous work [Lamballais *et al.*, 2017] has demonstrated the impact of workstations on throughput. Second, our optimization method only works for small 2D warehouses represented as a 4-neighbor grid. Future works can focus on generating larger 2D or 3D warehouses. Third, we can improve our method to optimize layouts with a larger agent density ( $P_a$ ). Overall, we are excited about the first proposed method for layout optimization and are looking forward to adapting our approach to more complex warehouse domains.

## Acknowledgments

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. In addition, this work was supported by the NSF CAREER Award (#2145077) and the CMU Manufacturing Futures Institute, made possible by the Richard King Mellon Foundation.

## References

- [Bhatt *et al.*, 2022] Varun Bhatt, Bryon Tjanaka, Matthew Fontaine, and Stefanos Nikolaidis. Deep surrogate assisted generation of environments. In *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, pages 37762–37777, 2022.
- [Cap *et al.*, 2015] Michal Cap, Jirı Vokrınek, and Alexander Kleiner. Complete decentralized method for on-line multi-robot trajectory planning in well-formed infrastructures. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 324–332, 2015.
- [Chen *et al.*, 2021] Zhe Chen, Javier Alonso-Mora, Xiaoshan Bai, Daniel D Harabor, and Peter J Stuckey. Integrated task assignment and path planning for capacitated multi-agent pickup and delivery. *IEEE Robotics and Automation Letters*, 6(3):5816–5823, 2021.
- [Contini and Farinelli, 2021] Antonello Contini and Alessandro Farinelli. Coordination approaches for multi-item pickup and delivery in logistic scenarios. *Robotics and Autonomous Systems*, 146:103871, 2021.
- [Cully *et al.*, 2015] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, May 2015.
- [Damani *et al.*, 2021] Mehul Damani, Zhiyao Luo, Emerson Wenzel, and Guillaume Sartoretti. PRIMAL<sub>2</sub>: Pathfinding via reinforcement and imitation multi-agent learning - lifelong. *IEEE Robotics and Automation Letters*, 6(2):2666–2673, 2021.
- [Erdmann and Lozano-Perez, 1986] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1419–1424, 1986.
- [Fontaine and Nikolaidis, 2021] Matthew C. Fontaine and Stefanos Nikolaidis. A quality diversity approach to automatically generating human-robot interaction scenarios in shared autonomy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2021.
- [Fontaine *et al.*, 2019] Matthew C. Fontaine, Scott Lee, L. B. Soros, Fernando De Mesentier Silva, Julian Togelius, and Amy K. Hoover. Mapping hearthstone deck spaces through map-elites with sliding boundaries. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 161–169, 2019.
- [Fontaine *et al.*, 2021a] Matthew C. Fontaine, Ya-Chuan Hsu, Yulun Zhang, Bryon Tjanaka, and Stefanos Nikolaidis. On the importance of environments in human-robot coordination. In *Proceedings of Robotics: Science and Systems (RSS)*, 2021.
- [Fontaine *et al.*, 2021b] Matthew C. Fontaine, Ruilin Liu, Ahmed Khalifa, Jignesh Modi, Julian Togelius, Amy K. Hoover, and Stefanos Nikolaidis. Illuminating Mario scenes in the latent space of a generative adversarial network. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 5922–5930, 2021.
- [Grenouilleau *et al.*, 2019] Florian Grenouilleau, Willem-Jan van Hoeve, and John N. Hooker. A multi-label A\* algorithm for multi-agent pathfinding. In *Proceedings of the International Conference on Automated Planning and Scheduling (ICAPS)*, pages 181–185, 2019.
- [Honig *et al.*, 2019] Wolfgang Honig, Scott Kiesel, Andrew Tinka, Joseph W. Durham, and Nora Ayanian. Persistent and robust execution of mapf schedules in warehouses. *IEEE Robotics and Automation Letters*, 4:1125–1131, 2019.
- [Justesen *et al.*, 2018] Niels Justesen, Ruben Rodriguez Torrado, Philip Bontrager, Ahmed Khalifa, Julian Togelius, and Sebastian Risi. Illuminating generalization in deep reinforcement learning through procedural level generation. In *NeurIPS Workshop on Deep Reinforcement Learning Workshop*, 2018.
- [Kou *et al.*, 2020] Ngai Meng Kou, Cheng Peng, Hang Ma, T. K. Satish Kumar, and Sven Koenig. Idle time optimization for target assignment and path finding in sortation centers. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 9925–9932, 2020.
- [Lamballais *et al.*, 2017] T. Lamballais, D. Roy, and M.B.M. De Koster. Estimating performance in a robotic mobile fulfillment system. *European Journal of Operational Research*, 256(3):976–990, 2017.
- [Li *et al.*, 2020] Jiaoyang Li, Kexuan Sun, Hang Ma, Ariel Felner, T. K. Satish Kumar, and Sven Koenig. Moving agents in formation in congested environments. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 726–734, 2020.
- [Li *et al.*, 2021] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W. Durham, T. K. Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 11272–11281, 2021.
- [Liu *et al.*, 2019] Minghua Liu, Hang Ma, Jiaoyang Li, and Sven Koenig. Task and path planning for multi-agent pickup and delivery. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1152–1160, 2019.
- [Ma *et al.*, 2017] Hang Ma, Jiaoyang Li, T. K. Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding for



- online pickup and delivery tasks. In *Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, pages 837–845, 2017.
- [Ma *et al.*, 2019] Hang Ma, Daniel Harabor, Peter J. Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 7643–7650, 2019.
- [Mouret and Clune, 2015] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *ArXiv*, abs/1504.04909, 2015.
- [Nguyen *et al.*, 2017] Van Nguyen, Philipp Obermeier, Tran Cao Son, Torsten Schaub, and William Yeoh. Generalized target assignment and path finding using answer set programming. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1216–1223, 2017.
- [Phillips and Likhachev, 2011] Mike Phillips and Maxim Likhachev. SIPP: Safe interval path planning for dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635, 2011.
- [Risi and Togelius, 2019] Sebastian Risi and Julian Togelius. Increasing generality in machine learning through procedural content generation. *Nature Machine Intelligence*, pages 1–9, 2019.
- [Shaker *et al.*, 2016] Noor Shaker, Julian Togelius, and Mark J. Nelson. *Procedural Content Generation in Games: A Textbook and an Overview of Current Research*. Springer, 2016.
- [Stern *et al.*, 2019] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Roman Barták, and Eli Boyarski. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pages 151–159, 2019.
- [Varambally *et al.*, 2022] Sumanth Varambally, Jiaoyang Li, and Sven Koenig. Which MAPF model works best for automated warehousing? In *Proceedings of the Symposium on Combinatorial Search (SoCS)*, pages 190–198, 2022.
- [Wan *et al.*, 2018] Qian Wan, Chonglin Gu, Sankui Sun, Mengxia Chen, Hejiao Huang, and Xiaohua Jia. Lifelong multi-agent path finding in a dynamic environment. In *Proceedings of the International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 875–882, 2018.
- [Wu *et al.*, 2020] Shasha Wu, Cheng Chi, Wei Wang, and Yaohua Wu. Research of the layout optimization in robotic mobile fulfillment systems. *International Journal of Advanced Robotic Systems*, 17(6):1729881420978543, 2020.
- [Wurman *et al.*, 2007] Peter R. Wurman, Raffaello D’Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 1752–1760, 2007.
- [Yang *et al.*, 2021] Xiuqing Yang, Xinglu Liu, Lijuan Feng, Jianquan Zhang, and Mingyao Qi. Non-traditional layout design for robotic mobile fulfillment system with multiple workstations. *Algorithms*, 14:203, 2021.
- [Zhang *et al.*, 2020] Hejia Zhang, Matthew C. Fontaine, Amy K. Hoover, Julian Togelius, Bistra Dilkina, and Stefanos Nikolaidis. Video game level repair via mixed integer linear programming. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE)*, 2020.
- [Zhang *et al.*, 2022] Yulun Zhang, Matthew C. Fontaine, Amy K. Hoover, and Stefanos Nikolaidis. Deep surrogate assisted map-elites for automated hearthstone deckbuilding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 158–167, 2022.