MULTI-OBJECTIVE LAYOUT OPTIMIZATION FOR MULTI-ROBOT COORDINATION IN AUTOMATED WAREHOUSE

Anonymous authors
Paper under double-blind review

ABSTRACT

We study the problem of multi-objective layout optimization for automated warehouses, where hundreds to thousands of robots are coordinated to transport packages from one location to another. Previous works have attempted to optimize the system throughput of warehouses by optimizing their layout (e.g., the physical locations of the storage shelves) while assuming a fixed storage capacity (i.e., the number of storage shelves). However, in practice, storage capacity also significantly affects the operating cost of the warehouse, because a smaller storage capacity requires the warehouse to occupy more land. Importantly, there are tradeoffs between the storage capacity and throughput, because a larger storage capacity creates more obstacles for robots, complicating the multi-robot coordination and potentially reducing throughput. In this paper, we generalize the state-of-the-art single-objective layout optimization methods based on Quality Diversity (QD) algorithms to multiple objectives, where we optimize for a Pareto front (PF) of throughput and storage capacity. We show that our optimized PF significantly outperforms vanilla multi-objective optimization algorithms across various robot task distributions, robot coordination algorithms, and layout sizes. We additionally present a Mixed Integer Linear Programming Solver (MILP) that enforces the optimized layout to be biconnected, which helps the downstream coordination algorithm move the robots more effectively. *Our code will be publicly available upon acceptance.*

1 INTRODUCTION

We study the problem of multi-objective layout optimization (MOLO) for automated warehouse with the objectives of maximizing throughput and storage capacity. With the flourish of e-commerce, automated warehouses operated by hundreds to thousands of robots are being increasingly used by major stakeholders such as Amazon (Brown, 2022), Walmart (Reuter, 2026), and JD.com (Wangshu, 2026) for efficient storage and operation. Although many prior works have attempted to improve warehouse throughput by solving the underlying Multi-Agent Path Finding (MAPF) problem (Stern et al., 2019) to more effectively coordinate robots (Ma et al., 2017; Liu et al., 2019; Okumura et al., 2019; Li et al., 2021; Chen et al., 2021; Damani et al., 2021; Varambally et al., 2022; Jiang et al., 2024), another line of work optimizes warehouse layout with fixed MAPF algorithms to improve throughput (Lamballais et al., 2017; Wu et al., 2020; Zhang et al., 2023b;a; Luo et al., 2026).

Although layout optimization has been shown to effectively improve throughput, state-of-the-art layout optimization methods assume a fixed, small storage capacity where merely 20% of the space is used for storage (Zhang et al., 2023b). In reality, the storage capacity can be much larger. For example, in prior research inspired by Amazon warehouses (Jiang et al., 2025) and Symbotic warehouses (Zheng et al., 2026), up to 45-50% of the warehouse space is used for storage. A higher storage capacity can reduce the operating cost of the warehouse.

Throughput and storage capacity are metrics that stakeholders are willing to optimize, but they form a set of competing goals for layout optimization. Increasing the storage capacity adds more storage shelves, which act as obstacles for the robots, creating a denser environment for robots coordination and reducing throughput due to traffic congestion. Therefore, it becomes natural to

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

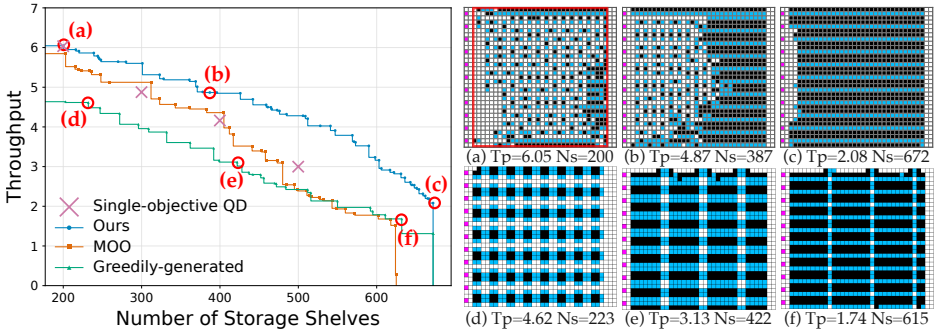


Figure 1: PF of layouts that optimizes throughput (*Tp*) and storage capacity (*Np*) comparing (1) our method, (2) multi-objective optimization (MOO) algorithm, (3) greedily-generated layouts by varying the size of the shelf block, and (4) single-objective results with various fixed numbers of shelves. Higher throughput and storage capacity (toward the upper right) are preferred. Robots move between randomly selected endpoints (blue) and workstations (pink) to transport packages.

consider the MOLO problem that searches for a Pareto front (PF) of layouts that optimizes both metrics. Practically, it gives warehouse stakeholders a set of high-quality layouts so they can choose based on the trade-off of efficiency and cost. Figure 1 gives an example PF found by our algorithm.

However, developing an effective MOLO algorithm poses challenges. First, state-of-the-art layout optimization methods exploit Quality Diversity (QD) algorithms to search for high-throughput layouts. They optimize the layouts by iteratively generating new ones and repairing them using a Mixed Integer Linear Programming (MILP) solver to enforce constraints, such as connectivity of the layout. They then evaluate the layouts with an expensive simulator that simulates the movement of the robots in the warehouse. Both the MILP solver and the simulator scale poorly with the size of the layout. Second, while prior works enforce the layout to be connected using MILP, such a constraint does not guarantee algorithmic completeness for some MAPF algorithms (Okumura et al., 2019).

In this paper, we generalize the state-of-the-art single-objective layout optimization method based on QD (Zhang et al., 2023a) to multiple objectives by incorporating Multi-Objective Quality Diversity (MOQD) algorithms. Specifically, we use MOQD to train a collection of Neural Cellular Automata (NCA) (Mordvintsev et al., 2020) layout generators capable of generating layouts of arbitrary sizes. To tackle the first challenge, we use a surrogate model to emulate the MILP solver and simulator to improve the sample efficiency of optimization. Inspired by prior works (Zhang et al., 2022; 2023b), the surrogate model is trained concurrently with the search instead of being pre-trained. To tackle the second challenge, we introduce a MILP solver that enforces the optimized layouts to be *biconnected*, which guarantees algorithmic completeness for some MAPF algorithms.

We make the following contributions: (1) we introduce a MOLO technique that optimizes for high throughput and storage capacity, offering layouts for stakeholders balancing operational efficiency and cost, (2) we introduce a MILP solver that enforces biconnectivity constraints on the optimized layouts, and (3) we show that our optimized layouts form superior PF compared to MOO algorithms as well as greedily-generated layouts across different MAPF algorithms, and layout sizes.

2 Problem Formulation

Following Zhang et al. (2023b), we first define warehouse layout and validity constraints. We then define objectives and the problem of MOLO.

Definition 1 (Warehouse Layout) *A warehouse layout is an undirected four-connected grid graph $G(V, E)$, where each tile can be one of workstation (pink), storage shelf (black), endpoint (blue), and empty space (white). Example layouts are shown in Figure 1. Shelves are not traversable and all other tile types are traversable. We define $\mathcal{L}$ as the space of all layouts. Goal locations of the robots are randomly sampled from workstations and endpoints alternatively.*

Definition 2 (Valid Layout) *A layout is valid iff (1) the corresponding graph of the layout is connected, and (2) at least one endpoint is adjacent to each shelf, and vise versa. We define $\mathcal{L}^* \subseteq \mathcal{L}$ as the space of all valid layouts.*

Definition 3 (Throughput) *Given a valid layout l, the number of robots R, and the number of timesteps T, throughput is obtained by running a simulation on l with R robots for T timesteps. A robot finishes a task once it arrives at its assigned goal location. Throughput computes the average number of goal locations assigned by all robots per timestep.*

Definition 4 (Storage Capacity) *Given a layout l, the storage capacity is defined as the number of storage shelves in the layout.*

Due to practical limits in relocating the workstations, we follow Zhang et al. (2023b) to divide the layout into a storage area and a non-storage area. The area inside the red box in Figure 1(a) is the storage area and the rest is the non-storage area. We optimize only the storage area.

Definition 5 (Multi-Objective Layout Optimization (MOLO)) *Given the layout of a non-storage area and a range of number of shelves $[N_s^-, N_s^+]$, the MOLO problem searches for valid layouts that simultaneously maximize throughput and storage capacity. Specifically:*

$$\max_{l \in \mathcal{L}^*} \quad \mathbf{f}(l) = (f_{tp}(l \mid R, T), f_{ns}(l)) \qquad \textit{s.t.} \qquad N_s^- \leq f_{ns}(l) \leq N_s^+. \tag{1}$$

$f_{tp} : \mathcal{L}^* \to \mathbb{R}$ computes the average throughput over N_e simulations, and $f_{ns} : \mathcal{L}^* \to \mathbb{R}$ computes the storage capacity. For two valid layouts l_1 and l_2, l_1 Pareto-dominates l_2 if $f_i(l_1) \geq f_i(l_2), \forall i \in \{tp, ns\}$, and the inequality is strict for at least one objective. A valid layout is Pareto optimal if it is not Pareto-dominated by any other valid layouts. The goal of MOLO is to find the set of Pareto-optimal layouts (i.e., a PF) representing trade-offs between throughput and storage capacity by optimizing *hypervolume* (HV). HV measures the volume of objective space dominated by a PF and bounded by a fixed reference point, with larger values indicating better objective quality.

3 Background and Related Work

3.1 Multi-Agent Path Finding

Multi-Agent Path Finding (MAPF) (Stern et al., 2019) aims to search for collision-free paths for multiple agents from their corresponding start to goal locations. In automated warehouses, lifelong MAPF (LMAPF) (Li et al., 2021), a variant of MAPF that continuously assigns new goals to agents, is widely adopted. Therefore, in this work, we adopt the classic LMAPF definition, where (1) robots move on a four-connected grid graph in discretized timesteps, (2) in each timestep, each robot can either move to an adjacent vertex or wait in the current vertex, and (3) two robots collide when they arrive at the same vertex or swap vertices at the same timestep.

The LMAPF algorithms have three main categories. Search-based methods (Wan et al., 2018; Li et al., 2021) decompose LMAPF into a sequence of MAPF problems and solve them one by one. They exhibit better solution quality than other methods but lack scalability. Rule-based methods (Wang & Botea, 2011; Okumura et al., 2019; Yukhnevich & Andreychuk, 2025; Jiang et al., 2026) move the robots step-by-step using pre-defined rules. They scale to more robots but have poor solution quality due to myopic planning. Learning-based methods use imitation learning (Jiang et al., 2025; Andreychuk et al., 2025) or multi-agent reinforcement learning (Damani et al., 2021; Wang et al., 2023) to train a neural network policy, but generally lack generality. In our experiments, we select RHCR (Li et al., 2021) and PIBT (Okumura et al., 2019) in our LMAPF simulator.

3.2 Multi-Objective and Quality Diversity Optimization

Multi-Objective Optimization. Multi-objective optimization (MOO) (Deb, 2011) simultaneously optimizes multiple objectives by finding a set of Pareto-optimal solutions representing different trade-offs among the objectives. Classic MOO algorithms have three main categories. Dominance-based algorithms, such as GSEMO (Laumanns et al., 2002; Giel, 2003) and NSGA-II (Deb et al., 2002), maintain a population of non-dominated solutions. Decomposition-based algorithms such

as MOEA/D (Zhang & Li, 2007) decompose the MO problem into a set of scalar optimization subproblems. Indicator-based algorithms such as SMS-EMOA (Beume et al., 2007) select solutions by their HV contributions. Beyond these families, MO-CMA-ES (Igel et al., 2007) extends the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) (Hansen, 2016) to multiple objectives by adapting a population of search strategies under multi-objective selection.

Quality Diversity. Quality Diversity (QD) (Mouret & Clune, 2015) simultaneously optimizes one objective and diversifies a set of diversity measure functions. Measure functions define a measure space. QD algorithms discretize the measure space into evenly spaced cells and search for the best solution in each cell. The discretized measure space is referred to as an *archive*. QD has been shown to help optimization escape local optima, both empirically (Zhang et al., 2026) and theoretically (Qian et al., 2024). Recently, the CMA-ES search kernel has been adapted to QD, inspiring CMA-ME (Vassiliades & Mouret, 2018) and CMA-MAE (Fontaine & Nikolaidis, 2023).

Multi-Objective Quality Diversity. MOQD algorithms (Pierrot et al., 2022; Zhao & Nikolaidis, 2025) generalize QD to multiple objectives, maintaining a set of non-dominated solutions in each cell instead of a single elite in the archive. MOME (Pierrot et al., 2022) maintains a population of solutions and updates the archive with NSGA-II-style non-dominated sorting. MOME-PGX (Janmohamed et al., 2023) combines gradient assistance and crowding exploration to improve the data efficiency of MOQD search. MO-CMA-MAE (Zhao & Nikolaidis, 2025) combines the CMA-ES search kernel with multi-objective selection to improve sample efficiency.

3.3 Warehouse Layout Optimization

Early works in operation research has explored the use of integer programming to optimize layouts while minimizing the distance between shelves and workstations (Yang et al., 2021). Other works use the queueing network to estimate the throughput of a given layout and guide the design of layouts (Lamballais et al., 2017; Wu et al., 2020), but do not consider interaction between multiple robots. Luo et al. (2026) explored using repeated runs of single-agent path finding algorithms to estimate robot traffic patterns and develops an annealing-based method to optimize layouts.

The state-of-the-art layout optimization techniques are based on QD. Zhang et al. (2023b) directly optimize the best object type at each vertex. To enforce validity, they use a MILP solver to repair the layouts. Finally, they evaluate the repaired layouts using a LMAPF simulator for throughput. To improve sample efficiency, they simultaneously train a surrogate model to emulate the MILP solver and the LMAPF simulator. However, the computational time scales exponentially with the size of the layout. Therefore, Zhang et al. (2023a) optimize neural network layout generators, namely Neural Cellular Automata (NCA) (Mordvintsev et al., 2020), to generate layouts of arbitrary sizes. Cellular Automata (CA) (Gardner, 1970) incrementally constructs a cell-based structure (such as a layout) through local interactions between cells. Each cell observes its neighboring cells and changes its state iteratively. NCA parameterizes the rules using a CNN that can be optimized. Zhang et al. (2023a) optimize NCA on small layout sizes and directly use it to generate larger layouts. All existing methods focus on single-objective optimization and tread throughput as the only objective. We attempt to generalize existing methods to multiple objectives.

4 Methods

We first formulate the MOLO problem as a MOQD problem and introduce our basic MOLO method based on MO-CMA-MAE (Zhao & Nikolaidis, 2025). We then introduce our advanced MOLO method by combining MO-CMA-MAE and prior single objective layout optimization methods (Zhang et al., 2023a;b). Finally, we introduce the MILP solver used to enforce validity and biconnectivity constraints on optimized layouts.

4.1 MOLO via MOQD and NCA

To formulate MOLO as a MOQD problem, in addition to the objectives $\mathbf{f}$, we also define a diversity measure function $\mathbf{m} : \mathcal{L}^* \to \mathbb{R}^m$ that returns the measure values that we wish to diversify, aiming to search for an archive of PFs, where each PF corresponds to a distinct range of measure values. At the end of the search, we aggregate all solutions in the archive, forming a global PF.

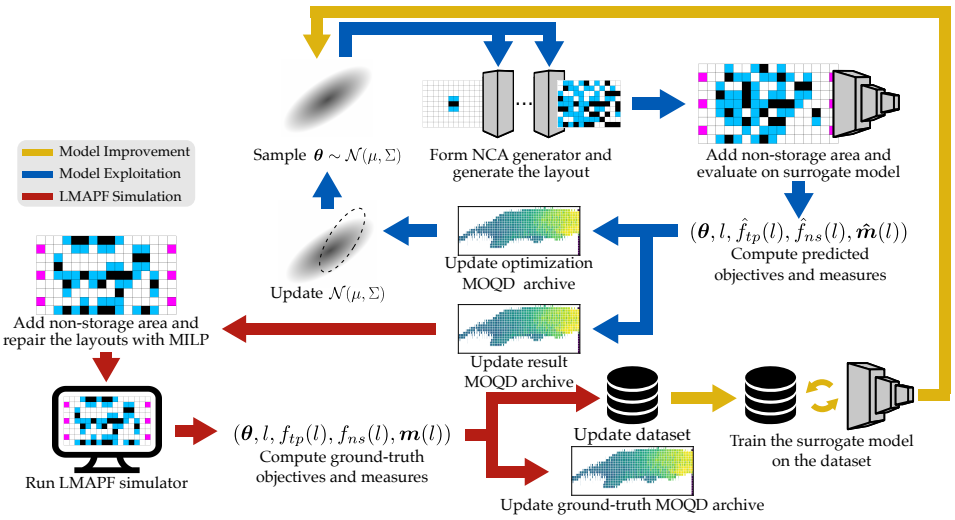


Figure 2: Overview of MO-DSAGE. We iterate over the model exploitation, model improvement, and LMAPF simulation phases until the number of evaluations in the LMAPF simulation phase reaches N_{eval}.

To search for the parameters of the NCA generators, we use MO-CMA-MAE (Zhao & Nikolaidis, 2025), which maintains a multi-variate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ and repeatedly samples it for new solutions. Starting from standard normal, we sample a batch of b parameter vectors and form b NCA generators. From a fixed initial layout l_0, we generate b layouts. We then repair the layouts to enforce validity and biconnectivity constraints by using the MILP solver, and evaluate the repaired layouts for objectives and measures. The outcome of the evaluation is used to update μ and Σ. MO-CMA-MAE runs until the number of evaluations reaches N_{eval}.

Objectives. Apparently, our objective is to optimize throughput f_{tp} and storage capacity f_{ns}. To allow the optimized NCA generators capable of generating layouts of arbitrarily large sizes, Zhang et al. (2023a) optimize $f'_{tp} = f_{tp} + \alpha\Delta$ instead of f_{tp}, where α is a hyperparameter. Δ is defined as the *similarity score* between the unrepaired and repaired layouts, quantifying the percentage of vertices that have the same tile types in the unrepaired and repaired layouts. By adding Δ to f_{tp}, we regularize the optimization to favor NCA generators that generate valid and biconnected layouts directly, instead of relying on the MILP solver to satisfy all these constraints. To coordinate the two objectives, we maintain two MOQD archives: an *optimization* archive that optimizes the regularized objective $\mathbf{f}' = (f'_{tp}, f_{ns})$ and a *result* archive that optimizes $\mathbf{f} = (f_{tp}, f_{ns})$. At the end of the search, we aggregate all solutions in the result archive, forming a global PF and return it as the result. In Appendix A.1, we discuss a variant of MO-CMA-MAE that explicitly optimize local and global PF.

Diversity Measures. We use *the layout entropy* (Lucas & Volz, 2019; Fontaine et al., 2021; Zhang et al., 2023a) as the diversity measure, which quantifies the degree of regularization of a given layout. The lower the layout entropy, the more regularized the layout patterns.

NCA Generator. We define an NCA generator as a function $g(l_0; \boldsymbol{\theta}, C)$:$\mathcal{L} \to \mathcal{L}$, where $l_0 \in \mathcal{L}$ is a fixed initial layout, $\boldsymbol{\theta}$ parameterizes the NCA, and $C \in \mathbb{Z}^+$ is the number of iterations that the NCA runs. We represent a 2D layout as a 3D tensor where each vertex of the grid is a one-hot encoded vector representing one of endpoint, shelf, and empty space. We use a 3-layer CNN with kernel sizes 3×3 with about 3,000 parameters as our NCA. The input and output of the NCA are layouts of the same sizes, allowing the same NCA to generate layouts of arbitrary sizes.

4.2 MO-DSAGE

Both the MILP solver and the LMAPF simulator are computationally expensive, making MO-CMA-MAE sample inefficient. Deep Surrogate Assisted Generation of Environment (DSAGE) (Bhatt et al., 2022) adopts the idea of model-based QD (Zhang et al., 2022) and trains a surrogate model

concurrently with the QD search to improve sample efficiency. We extend DSAGE to the multi-objective domain, introducing MO-DSAGE to solve MOLO.

Figure 2 shows an overview of MO-DSAGE. We start by sampling N_{init} random NCA generators, and then generate/repair/evaluate N_{init} initial layouts for the regularized objectives $\mathbf{f}'$ and measures $\mathbf{m}$. These data are added to a dataset and used to train the surrogate model. We then enter the *model exploitation* (blue) phase where we run MO-CMA-MAE (Section 4.1) except that we use the surrogate model to predict $\mathbf{f}'$ and $\mathbf{m}$. The model exploitation phase runs for N_{inner} surrogate evaluations. The layouts in the result archive are those predicted by the surrogate model to have high throughput and storage capacity, and are thus evaluated in simulations. Therefore, we downsample the result MOQD archive with at most N_{select} layouts (Appendix A.2) and repair/evaluate them during the *LMAPF simulation* (red) phase. The evaluated layouts are then added to the dataset and a separate ground-truth MOQD archive to optimize unregularized objectives $\mathbf{f}$. Finally, we further train the surrogate model with the updated dataset (yellow), starting a new iteration. The search continues until the number of evaluations in LMAPF simulation phase reaches N_{eval}. The PF formed by solutions stored in the ground-truth archive are returned as the result.

4.3 Enforcing Validity and Biconnectivity Constraints

After the NCA generates a layout, we add the non-storage area and repair it so that it becomes biconnected and valid. Zhang et al. (2023b) have presented a MILP solver that enforces the validity constraint. We adopt and expand their formulation so that the repaired layouts are biconnected. We use (u, v) to represent a directed edge from u to v and $\{u, v\}$ a undirected edge between u and v. Given an arbitrary layout and its corresponding undirected graph $G(V, E)$, we first form a bidirected graph $G'(V, E')$, where $E' = \bigcup_{\{u,v\}\in E}\{(u, v), (v, u)\}$. We define $O = \{w, s, e, p\}$ as the set of all tile types, including the workstation w, the shelf s, the endpoint e and the empty space p. To repair a layout, we search for a matching between each vertex $v \in V$ and each tile type $o \in O$.

Objective. We repair the layout with the objective of minimizing the hamming distance between the unrepaired layout and the repaired one. For $v \in V$, let a binary scalar x_v^0 indicate whether vertex v is tile type x in the unrepaired layout. Then, our objective is:

$$\min \sum_{v\in V} \sum_{x\in O} |x_v - x_v^0|. \tag{2}$$

For the constraints, we first discuss the constraints that enforce the layout for validity and biconnectivity. Trivially, if a layout is biconnected, it is connected. Therefore, we have (1) tile-type uniqueness constraints, (2) shelf-endpoint adjacency constraints, and (3) biconnectivity constraints. In addition, we have warehouse domain-specific constraints, including (1) non-storage area constraints and (2) number of shelves constraint, which are discussed in Appendix A.3.

Tile Type Uniqueness Constraints. We require that each vertex have exactly one tile type:

$$\forall v \in V,\ w_v + e_v + s_v + p_v = 1. \tag{3}$$

Shelf-Endpoint Adjacency Constraints. We require the shelves and endpoints to be adjacent:

$$\forall v \in V,\ \sum_{u:(u,v)\in E} s_u \geq e_v, \qquad \sum_{u:(u,v)\in E} e_u \geq s_v \tag{4}$$

The two inequality in Equation (4) enforces that (1) each endpoint is adjacent to at least one shelf and (2) each shelf is adjacent to at least one endpoint. Taking the first inequality as an example, the left side computes the number of adjacent shelves of vertex v. If it is 0, v cannot be an endpoint. Otherwise, v can be an endpoint. The second inequality is analogical.

Biconnectivity Constraint. We require the repaired layout to be biconnected because many MAPF algorithms requires biconnectivity to guarantee algorithmic completeness (Okumura et al., 2019; Yukhnevich & Andreychuk, 2025).

Definition 6 (Biconnected Layout) *A layout is biconnected if the underlying undirected graph $G(V, E)$ has no bridges in E. An edge $\{u, v\} \in E$ is a bridge if removing it disconnects G.*

Robbins (1939) has established that the edge directions can be assigned to an undirected graph G so that G becomes strongly connected if and only if G is biconnected. Based on this claim, Duhamel & Santos (2024) have presented a MILP formulation that assigns edge directions to the edges of a undirected graph so that it becomes strongly connected. In our case, however, we are repairing a layout (and thus its underlying graph) so that it becomes biconnected.

According to Robbins (1939), given a layout with its underlying undirected graph $G(V, E)$, if we can assign edge directions to E so that the transformed directed graph $G^R(V, E^R)$ becomes strongly connected, G is biconnected, and therefore the layout is biconnected. Importantly, if G^R is strongly connected, then (1) on G^R, there is a path from a random vertex $v \in V$ to all other vertices, and (2) on another graph $G^B(V, E^B)$, where the edge directions of E^R are reversed (i.e., $E^B = \bigcup_{\forall(u,v)\in E^R}\{(v, u)\}$), there is a path from v to all other vertices (Sharir, 1981). Therefore, there are always spanning trees T^R of G^R and T^B of G^B, respectively, where v is the root and v can reach all other vertices. Our MILP is formulated to find these two spanning trees by using network flow.

For each undirected edge $\{u, v\} \in E$, we define binary variables x_{uv}, x_{vu}, and $x_{\{u,v\}}$, as well as non-negative integer flow variables f^R_{uv}, f^R_{vu}, f^B_{uv}, and f^B_{vu}. x_{uv} and x_{vu} indicate whether (u, v) and (v, u) are chosen, and $x_{\{u,v\}}$ indicates whether there is an edge between u and v. The flow variable f^R_{uv} defines the amount of flow on (u, v) on the spanning trees T^R. Other flow variables are analogous. We first require:

$$\forall\{u, v\} \in E, x_{\{u,v\}} \leq 1 - s_u, \quad x_{\{u,v\}} \leq 1 - s_v, \quad x_{\{u,v\}} \geq 1 - s_u - s_v, \quad x_{uv} + x_{vu} = x_{\{u,v\}}. \tag{5}$$

Recall that $s_u = 1$ iff u is a shelf and thus an obstacle. Therefore, Equation (5) ensure that (1) only one of (u, v) and (v, u) is selected, (2) if neither u nor v are shelves ($s_u = s_v = 0$), then there exists an edge between u and v ($x_{\{u,v\}} = 1$) and we can select between (u, v) and (v, u) ($x_{uv} + x_{vu} = 1$).

We select a vertex $v_0 \in V$ as the root in both T^R and T^B. Let $K = |V| - \sum_v s_v$ be the number of traversable vertices, $M = |V| - 1$ be the number of non-root vertices. We define the constraints:

$$\forall(u, v) \in E', 0 \leq f^R_{uv} \leq M x_{uv}, \qquad 0 \leq f^B_{uv} \leq M x_{vu}\ , \tag{6}$$

$$\sum_{u:(v_0,u)\in E'} f^R_{v_0u} = K - 1, \qquad \sum_{u:(v_0,u)\in E'} f^B_{v_0u} = K - 1. \tag{7}$$

For every $v \neq v_0$, the flows satisfy, respectively,

$$(1 - s_v) + \sum_{u:(v,u)\in E'} f^R_{vu} = \sum_{u:(u,v)\in E'} f^R_{uv}, \quad (1 - s_v) + \sum_{u:(v,u)\in E'} f^B_{vu} = \sum_{u:(u,v)\in E'} f^B_{uv}. \tag{8}$$

Conceptually, we ask the outgoing edges of the root v_0 to supply $K - 1$ units of flow, and each other vertex absorbs 1 unit in both spanning trees. Equation (6) defines the lower and upper limits of the flow. Equation (7) defines the flow on the outgoing edges of the root vertex v_0. Equation (8) defines the flow conservation in all other non-root vertices. The term $(1 - s_v)$ ensures that if v is a shelf, v does not absorb any flow. Note that v_0 is required to be a non-shelf vertex. In our setup, we choose a fixed vertex in the non-storage area as v_0 because the non-storage area does not have shelves.

5 Experiments

5.1 Experiment Setup

Warehouse Setup. We optimize NCA generators with layouts of size $S_1 = 36 \times 33$. The number of shelves is constrained to $[N_s^-, N_s^+] = [200, 672]$. To validate the necessity of optimizing biconnected layouts, we consider variants of all methods that enforce the layouts to be valid. We give

PIBT				RHCR			
Valid		Valid and Biconnected		Valid		Valid and Biconnected	
Algorithm	HV ↑	Algorithm	HV ↑	Algorithm	HV ↑	Algorithm	HV ↑
MO-DSAGE	1944.5 ± 176.7	**MO-DSAGE**	$\mathbf{2138.1 \pm 53.1}$	MO-DSAGE	1837.2 ± 166.7	**MO-DSAGE**	$\mathbf{1873.3 \pm 140.4}$
MO-CMA-MAE	1419.2 ± 121.8	MO-CMA-MAE	1560.8 ± 107.5	MO-CMA-MAE	1164.2 ± 35.3	MO-CMA-MAE	1203.4 ± 71.3
CMA-MOA	1469.5 ± 71.6	CMA-MOA	1542.8 ± 128.2	CMA-MOA	1256.6 ± 122.9	CMA-MOA	1256.1 ± 96.9
SO-QD	—	SO-QD	1052.9 ± 222.9	SO-QD	—	SO-QD	—
Greedy	1394.0	Greedy	1389.8	Greedy	1235.1	Greedy	868.4

Table 1: Hypervolumes (HVs) of our methods comparing with the baseline with both PIBT and RHCR. Results are reported in the form of $x \pm y$ where x is the mean and y is the 95% Student-t confidence-interval half-widths over four runs.

all MILP solvers a time limit of 150 seconds. If the MILP solver cannot find a solution, the layout would not be included in the update of the covariance matrix.

LMAPF Simulation. We use PIBT (Okumura et al., 2019) and RHCR (Li et al., 2021). For RHCR, we use PBS (Ma et al., 2019) as the MAPF planner and SIPP (Phillips & Likhachev, 2011) as the single-agent planner, and we replan for 10-step paths every 5 timesteps. Throughput is averaged over $N_e = 5$ simulations of $T = 1{,}000$ timesteps per layout. Since shelves are obstacles and the number of shelves varies, we set the number of robots to $R = \lfloor P_A(|V|-N_s) \rfloor$, where $|V|-N_s$ is the number of traversable vertices, and P_A is the density of robots on free vertices. We use $P_A = 0.33$ for PIBT and $P_A = 0.2$ for RHCR. We use a lower P_A for RHCR because it is more likely to have deadlocks, causing traffic congestion. Appendix B.1 presents computational resources.

Baselines. We consider four baselines: (1) MO-CMA-MAE (Zhao & Nikolaidis, 2025), the state-of-the-art MOQD algorithm without surrogate modeling, (2) CMA-MOA, a MOO algorithm that does not consider diversity measures, (3) single-objective QD (SO-QD) using CMA-MAE (Fontaine & Nikolaidis, 2023), and (4) a set of greedily-generated layouts. CMA-MOA follows the same optimization procedure as MO-CMA-MAE, except that its measure space is zero-dimensional. Thus, its archive reduces to a single cell shared by all candidates. SO-QD optimizes throughput alone in separate CMA-MAE searches with fixed shelf counts $N_s \in \{200, 300, 400, 500, 600, 672\}$. Due to limit in compute, we only run it with PIBT and biconnectivity MILP. The greedily-generated layouts are designed by varying the sizes of the blocks of the shelves and tiling the blocks and applying MILP solvers to repair them. Figure 1(d)-(f) shows examples and Appendix B.2 presents the pseudocode.

Diversity Measure and Archive. All QD and MOQD methods use layout entropy as the sole diversity measure. The archive has 50 equal-width entropy bins over $[0, 1]$. We store 50 candidates per cell and thus 2,500 candidates in the MOQD archives in total.

Search Settings. MO-DSAGE uses $N_{\text{init}} = 500$, $N_{\text{inner}} = 5{,}000$, and $N_{\text{select}} \leq 250$. MO-DSAGE, MO-CMA-MAE, and CMA-MOA each use $N_{\text{eval}} = 10{,}000$. SO-QD runs $N_{\text{eval}} = 2,000$ for each N_s, resulting in a total of 12,000 evaluations. For C, MO-CMA-MAE, CMA-MOA and SO-QD use $C = 200$ and MO-DSAGE uses $C = 50$. For b, MO-DSAGE and MO-CMA-MAE use $b = 80$ and CMA-MOA uses $b = 50$. Appendix B.3 presents details of surrogate model training.

Scaling To Larger Layouts. We can directly use the optimized NCA to generate larger layouts without further optimization. We use the NCAs to generate layouts of sizes $S_2 = 69 \times 69$ and $S_3 = 101 \times 102$. We then repair them with the biconnectivity MILP and run 10 LMAPF simulations to get the average throughput. We compare all algorithms with PIBT. We use $C = 200$ and $C = 300$ for S_2 and S_3, respectively. Detailed setups are given in Appendix B.4.

5.2 Experimental Results

PF of Our Methods vs. Baselines. We present the hypervolume of the optimized PF in Table 1, with the optimization curve over 10k evaluations averaged over 4 runs shown in Figure 3. The results show that our optimized PFs with MO-DSAGE have significantly better hypervolume than the baselines, with a gap of 32.3–49.1%. The layouts shown in Figure 1 has demonstrated that the NCAs learn to put non-traversable shelves away from the workstations so that robots have more room to coordinate around the workstations, where the traffic accumulates. However, directly applying MO-CMA-MAE does not demonstrate significant advantage compared to that of CMA-MOA.

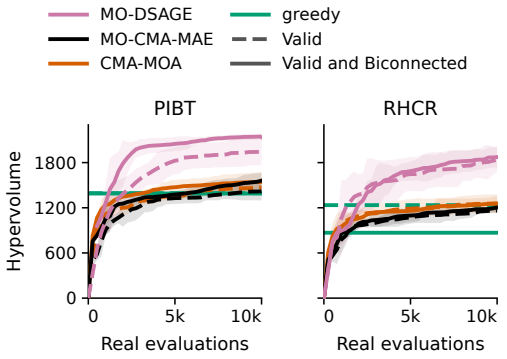


Figure 3: Optimization curves over 10k evaluations. Shading shows 95% confidence intervals.

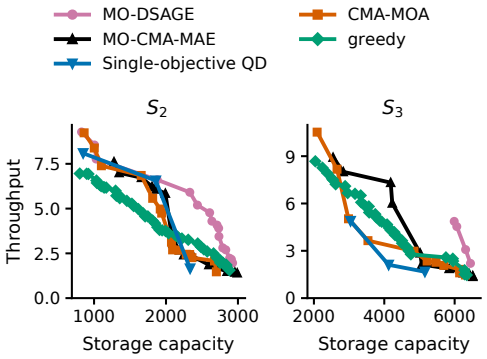


Figure 4: Scaling to $S_2 = 69 \times 69$ and $S_3 = 101 \times 102$ with PIBT and biconnectivity MILP.

	MO-DSAGE	MO-CMA-MAE	CMA-MOA	Greedy	Single-objective QD
S_2 HV ↑	**12,161.4**	9,868.2	9,153.5	8,643.7	7,409.0
S_3 HV ↑	21,425.2	**23,287.9**	17,421.3	20,976.5	9,715.4

Table 2: Hypervolume of Pareto fronts for layouts of sizes S_2 and S_3.

Validity vs. Biconnectivity MILP. Optimizing the layouts while preserving biconnectivity generally leads to better performance than only enforcing the layouts to be valid. The advantage is clearer in PIBT than in RHCR because (1) PIBT is proven to be complete in biconnected graphs (Okumura et al., 2019) and (2) RHCR plans for a longer horizon than PIBT, providing robots more chance to escape from non-biconnected components of the graph. Interestingly, the biconnectivity MILP does not improve the hypervolume of greedily-generated layouts. We conjecture that the way we generate these layouts made it inherently hard for the biconnectivity MILP to return a solution within our given time budget, resulting in many layouts being abolished.

Scaling To Larger Layouts. We evaluates NCA generators optimized on the small layout to S_2 and S_3. Table 2 shows the PFs and Table 2 shows the hypervolumes. At the size of S_2, MO-DSAGE achieves the largest hypervolume, being 23.2% higher than MO-CMA-MAE and 40.7% higher than the greedy baseline. However, transfer is less consistent with S_3: MO-DSAGE retains high-capacity solutions, including a layout with 5,988 shelves and throughput 4.87, but MO-CMA-MAE attains a larger hypervolume and CMA-MOA reaches a higher maximum throughput of 10.52. We conjecture that the MILP solver returns bad repair for MO-DSAGE with S_3 (details are discussed in Appendix C.4).

Additional Experiment Results. Appendix C.1 shows results under a different warehouse setup. Appendix C.2 compares MO-DSAGE with a variant of it that considers global hypervolume improvement (HVI). Appendix C.3 shows ablations on different diversity measure functions.

6 Conclusion

In this paper, we introduce MO-DSAGE, a surrogate-assisted multi-objective layout optimizer that tunes a collection of NCA generators to produce Pareto-optimal layouts maximizing throughput and storage capacity. We also propose a MILP solver to enforce biconnectivity. Experiments show improved hypervolume over a MOO baseline and state-of-the-art MOQD methods, with biconnectivity repair particularly beneficial for completeness-guaranteeing LMAPF solvers, and demonstrate zero-shot transfer to layouts $\times 2$ and $\times 3$ larger than those seen during optimization.

Our work has many limitations. First, we can incorporate the optimization of workstations as (Lamballais et al., 2017) has demonstrated the impact of workstations on throughput. Second, MO-DSAGE remains computationally demanding, requiring 48 hours to obtain the results shown in the paper. Future works can investigate further improving the sample efficiency of MO-DSAGE.

AI USE STATEMENT

In this work, we **have** used generative AI tools to implement methods, support qualitative and thematic data analysis, interpret results, create or modify scientific figures or images, edit a research paper to improve readability, and propose a title or keywords for a research paper. We reviewed all AI-assisted work. We checked the LLM-edited texts and modified them if they are inaccurate or unclear. The LLM-generated code was verified and the authors tested its correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

We **have NOT** used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, clean and reformat dataset, summarize or analyze existing literature, draft parts of a research paper.

REPRODUCIBILITY STATEMENT

To ensure reproducibility of our work, we have: (1) included detailed explanation of the proposed methods in Section 4 and Appendix A, (2) discussed detailed hyper-parameters and computational resources used to conduct the experiments in Section 5.1 and Appendix B, and (3) included source code along with a detailed README to reproduce the results in the supplemental material. The source code will be publicly available upon acceptance of this paper.

ETHICAL STATEMENT

This work studies multi-objective layout optimization for automated warehouses. Although optimized layouts may increase operational efficiency, throughput and storage capacity do not capture worker well-being, accessibility, or physical safety. The constraints enforced by the MILP solver and the simulated performance do not establish suitability for real-world deployment, which would require additional safety validation and consultation with affected workers. Warehouse automation may also affect employment and work intensity. We acknowledge these potential downstream impacts.

REFERENCES

Anton Andreychuk, Konstantin Yakovlev, Aleksandr Panov, and Alexey Skrynnik. MAPF-GPT: Imitation learning for multi-agent pathfinding at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 23126–23134, 2025.

Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

Varun Bhatt, Bryon Tjanaka, Matthew Fontaine, and Stefanos Nikolaidis. Deep surrogate assisted generation of environments. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 37762–37777, 2022.

Varun Bhatt, Heramb Nemlekar, Matthew Christopher Fontaine, Bryon Tjanaka, Hejia Zhang, Ya-Chuan Hsu, and Stefanos Nikolaidis. Surrogate assisted generation of human-robot interaction scenarios. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 513–539. PMLR, 06–09 Nov 2023.

Alan S. Brown. How Amazon robots navigate congestion. `https://www.amazon.science/latest-news/how-amazon-robots-navigate-congestion`, 2022. Accessed: 2023-05-09.

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.

Mehul Damani, Zhiyao Luo, Emerson Wenzel, and Guillaume Sartoretti. PRIMAL$_2$: Pathfinding via reinforcement and imitation multi-agent learning - lifelong. *IEEE Robotics and Automation Letters*, 6(2):2666–2673, 2021.

Kalyanmoy Deb. Multi-objective optimisation using evolutionary algorithms: an introduction. In *Multi-objective evolutionary optimisation for product design and manufacturing*, pp. 3–34. Springer, 2011.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Christophe Duhamel and Andréa Cynthia Santos. The strong network orientation problem. *International Transactions in Operational Research*, 31(1):192–220, 2024.

Matthew Fontaine and Stefanos Nikolaidis. Covariance matrix adaptation map-annealing. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 456–465, 2023.

Matthew C. Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K. Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 94–102, 2020. ISBN 9781450371285. doi: 10.1145/3377930.3390232. URL https://doi.org/10.1145/3377930.3390232.

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. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pp. 5922–5930, 2021.

Martin Gardner. Mathematical games – the fantastic combinations of john conway's new solitaire game "life". *Scientific American*, 223(4):120–123, 1970.

Oliver Giel. Expected runtimes of a simple multi-objective evolutionary algorithm. In *Proceedings of the 2003 Congress on Evolutionary Computation (CEC)*, volume 3, pp. 1918–1925, 2003.

Nikolaus Hansen. The CMA evolution strategy: A tutorial. *ArXiv*, abs/1604.00772, 2016. URL https://api.semanticscholar.org/CorpusID:15038271.

Christian Igel, Nikolaus Hansen, and Stefan Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

Hannah Janmohamed, Thomas Pierrot, and Antoine Cully. Improving the data efficiency of multi-objective quality-diversity through gradient assistance and crowding exploration. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 165–173, 2023.

He Jiang, Yulun Zhang, Rishi Veerapaneni, and Jiaoyang Li. Scaling lifelong multi-agent path finding to more realistic settings: Research challenges and opportunities. In *Proceedings of the International Symposium on Combinatorial Search (SoCS)*, pp. 234–242, 2024. doi: 10.1609/socs.v17i1.31565.

He Jiang, Yutong Wang, Rishi Veerapaneni, Tanishq Harish Duhan, Guillaume Adrien Sartoretti, and Jiaoyang Li. Deploying ten thousand robots: Scalable imitation learning for lifelong multi-agent path finding. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–7, 2025.

Zixiang Jiang, Yulun Zhang, Rishi Veerapaneni, and Jiaoyang Li. Planning over mapf agent dependencies via multi-dependency pibt. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tim Lamballais, Debjit Roy, and René M. B. M. de Koster. Estimating performance in a robotic mobile fulfillment system. *Eur. J. Oper. Res.*, 256:976–990, 2017.

Marco Laumanns, Lothar Thiele, Eckart Zitzler, Emo Welzl, and Kalyanmoy Deb. Running time analysis of multi-objective evolutionary algorithms on a simple discrete optimization problem. In *Parallel Problem Solving from Nature (PPSN VII)*, volume 2439 of *Lecture Notes in Computer Science*, pp. 44–53. Springer, 2002.

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)*, pp. 11272–11281, 2021.

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)*, pp. 1152–1160, 2019.

Simon M. Lucas and Vanessa Volz. Tile pattern KL-divergence for analysing and evolving game levels. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 170–178, 2019. ISBN 9781450361118. doi: 10.1145/3321707.3321781. URL https://doi.org/10.1145/3321707.3321781.

Xiangjie Luo, Yulun Zhang, Miyuki Koshimura, Makoto Yokoo, and Jiaoyang Li. Stress-relief annealing: Polynomial-time simulation-free layout optimization for automated warehouses. *arXiv*, abs/2608.01024, 2026.

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 Multiagent Systems (AAMAS)*, pp. 837–845, 2017.

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)*, pp. 7643–7650, 2019.

Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 2020. doi: 10.23915/distill.00023.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *ArXiv*, abs/1504.04909, 2015.

Keisuke Okumura, Manao Machida, Xavier Défago, and Yasumasa Tamura. Priority inheritance with backtracking for iterative multi-agent path finding. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 535–542, 2019.

M. Phillips and M. Likhachev. SIPP: Safe interval path planning for dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5628–5635, 2011.

Thomas Pierrot, Guillaume Richard, Karim Beguir, and Antoine (GECCO) Cully. Multi-Objective Quality Diversity Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 139–147, 2022. URL http://arxiv.org/abs/2202.03057.

Chao Qian, Ke Xue, and Ren-Jian Wang. Quality-diversity algorithms can provably be helpful for optimization. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 6994–7002, 2024.

Dominick Reuter. Walmart's ai-warehouses slash the time it takes store workers to unload trucks. https://www.businessinsider.com/walmarts-ai-warehouses-slash-time-for-workers-to-unload-trucks-2026-6, June 2026. Accessed: 2026-08-11.

H. E. Robbins. A Theorem on Graphs, with an Application to a Problem of Traffic Control. *The American Mathematical Monthly*, 46(5):281, May 1939. ISSN 00029890. doi: 10.2307/2303897. URL http://www.jstor.org/stable/2303897?origin=crossref.

Micha Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, 7(1):67–72, 1981.

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)*, pp. 151–159, 2019.

Sumanth Varambally, Jiaoyang Li, and Sven Koenig. Which MAPF model works best for automated warehousing? In *Proceedings of the Symposium on Combinatorial Search (SoCS)*, pp. 190–198, 2022.

Vassiiis Vassiliades and Jean-Baptiste Mouret. Discovering the elite hypervolume by leveraging interspecies correlation. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 149–156, 2018. ISBN 9781450356183.

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)*, pp. 875–882, 2018.

K. Wang and A. Botea. MAPP: A scalable multi-agent path planning algorithm with tractability and completeness guarantees. *Journal of Artificial Intelligence Research*, 42:55–90, 2011.

Yutong Wang, Bairan Xiang, Shinan Huang, and Guillaume Sartoretti. Scrimp: Scalable communication for reinforcement- and imitation-learning-based multi-agent pathfinding. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9301–9308, 2023.

Luo Wangshu. Logistics growth driven by smart systems. `https://mobile.chinadaily.com.cn/cn/html5/2026-02/06/content_006_6984e34fed50ccabe1521ad1.htm`, February 2026. Accessed: 2026-08-11.

Shasha Wu, Cheng Chi, Wei Wang, and Yao Wu. Research of the layout optimization in robotic mobile fulfillment systems. *International Journal of Advanced Robotic Systems*, 17, 2020.

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.

Egor Yukhnevich and Anton Andreychuk. Enhancing PIBT via multi-action operations. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2025.

Qingfu Zhang and Hui Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

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)*, pp. 158–167, 2022. ISBN 9781450392372. doi: 10.1145/3512290.3528718.

Yulun Zhang, Matthew C. Fontaine, Varun Bhatt, Stefanos Nikolaidis, and Jiaoyang Li. Arbitrarily scalable environment generators via neural cellular automata. In *Proceedings of the Advances in Neural Information Processing Systems (NeurIPS)*, pp. 57212–57225, 2023a.

Yulun Zhang, Matthew C. Fontaine, Varun Bhatt, Stefanos Nikolaidis, and Jiaoyang Li. Multi-robot coordination and layout design for automated warehousing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 5503–5511, 2023b.

Yulun Zhang, Varun Bhatt, Matthew C. Fontaine, Stefanos Nikolaidis, and Jiaoyang Li. Optimization of edge directions and weights for mixed guidance graphs in lifelong multi-agent path finding. *arXiv*, abs/2602.23468, 2026.

Shihan Zhao and Stefanos Nikolaidis. Multi-Objective Covariance Matrix Adaptation MAP-Annealing. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 673–682, 2025.

Han Zheng, Yining Ma, Brandon Araki, Jingkai Chen, and Cathy Wu. Learning-guided Prioritized Planning for Lifelong Multi-Agent Path Finding in Warehouse Automation. *Journal of Artificial Intelligence Research*, 85, 2026.

A Additional Algorithms

A.1 Incorporating Global HV Improvement in MO-CMA-MAE

In Section 4.1, we discussed using MO-CMA-MAE (Zhao & Nikolaidis, 2025) to solve MOLO. However, as an MOQD algorithm, MO-CMA-MAE only optimizes for a *local* PF in each cell of the MOQD archive, instead of a *global* PF over the entire search space. Therefore, we explore if incorporating the optimization of global PF can improve the optimization outcome of MOLO.

We present MO-CMA-MAE-HVI, a diversity-assisted MOLO method based on CMA-ME (Fontaine et al., 2020). Following MOME (Pierrot et al., 2022), we discretize the behavior space into cells and store multiple solutions in each cell. However, instead of storing not only the Pareto set, we also retain solutions from the top-ranked fronts according to their non-dominated sorting, within the per-cell capacity K. To guide CMA-ES in this MOQD search, we replace the scalar in-cell improvement used by CMA-ME with a combination of local and global hypervolume improvements. For an offspring x with objective vector $\boldsymbol{f}$ assigned to cell c, we define its local hypervolume improvement as the improvement of hypervolume of the cell when the solution is inserted into the cell, i.e., $\Delta^{\text{cell}} = HV(\mathcal{F}_c \cup \{\boldsymbol{f}\}) - HV(\mathcal{F}_c)$, where $\mathcal{F}_c$ denotes the objective vectors of the existing solutions in cell c. To make the CMA-ES aware of the global MO optimization and explicitly promote the global multi-objective performance, we further define the global hypervolume improvement $\Delta^{\text{global}} = HV(\mathcal{F} \cup \{\boldsymbol{f}\}) - HV(\mathcal{F})$ as the improvement of hypervolume of the global archive when the solution is inserted into the archive, where $\mathcal{F}$ contains the objective vectors of all solutions in the archive. We then rank the offspring using the combined improvement $\Delta = \Delta^{\text{cell}} + \lambda\Delta^{\text{global}}$ and use it to update the CMA-ES distribution. When no offspring survives, we restart CMA-ES by sampling its new mean μ uniformly at random from the global PF of the archive, since these solutions are of higher quality and may direct the search toward a globally competitive region.

The details of the method are shown in Algorithm 1. The method first initializes an empty MOQD archive $\mathcal{A}$ and the CMA-ES distribution in line 1. At each iteration, the method samples a batch of N offspring NCA parameter vectors x_i from the current Gaussian distribution in line 4. For each parameter vector, the method generates a layout from them, repairs the layout with MILP, evaluates its objective vector $\boldsymbol{f}_i$ and diversity measur $\boldsymbol{m}_i$ by running the LMAPF simulator in line 6–8. The cell c_i corresponding to each offspring is decided by the diversity measur $\boldsymbol{m}_i$ in line 9. After that, the method computes the improvement Δ_i of each offspring by calculating the improvements to the hypervolume of both the local cell and the global archive when the offspring is inserted in line 12–13, and uses their weighted sum as the final improvement in line 14. Then, the method sorts the offspring according to the final improvements Δ_i in descending order in line 15, and uses it to update the CMA-ES distribution in line 16. In line 17–19, each cell $\mathcal{A}_c$ of the archive $\mathcal{A}$ is updated by applying NSGA-II survivor selection to existing solutions and the offspring assigned to that cell with capacity K. If none of the offspring survives, the method restarts CMA-ES with its μ uniformly randomly selected from the global PF of the archive $\mathcal{A}$ in line 21. The method iterates until T generations, and returns the final archive $\mathcal{A}$ in the end.

MO-DSAGE-HVI. Similar to MO-CMA-MAE, MO-CMA-MAE-HVI is bottlenecked by the slow MILP solver and LMAPF simulator. Therefore, we follow the same procedure as Section 4.2 to develop MO-DSAGE-HVI, which is the same as MO-DSAGE except that the model exploitation phase runs MO-CMA-MAE-HVI instead of MO-CMA-MAE.

A.2 MOQD Archive Downsampling

Both MO-DSAGE and MO-DSAGE-HVI downsample the surrogate result archive before repair/evaluate the layouts with the MILP solver and LMAPF simulator during the LMAPF simulation phase. Bhatt et al. (2022; 2023) established that downsampling after the model exploitation phase of DSAGE can reduce the number of evaluations we run during the LMAPF simulation phase so that the surrogate model can be sufficiently trained. A successful downsampling shall retain coverage of the measure space and the storage-capacity objective while limiting each real-evaluation phase to at most 250 NCA generators by coarsening the second axis of the objective space, namely the number of shelves. If necessary, the diversity measure space is also coarsened. The per-cell solution-capacity axis is not a diversity-measure axis and is not coarsened.

Algorithm 1 MO-CMA-MAE-HVI

Input: number of iterations T, batch size N, discretized behavior space $\mathcal{M}$, per-cell capacity K, coefficient of global HV improvement λ
Output: MOQD archive $\mathcal{A}$ of NCA generator parameters

Initialize $\mathcal{A} \leftarrow \emptyset$ and the CMA-ES state (μ, Σ);
$t \leftarrow 1$;
while $t \leq T$ **do**
  ▷ *Sample offspring solutions (NCA parameters)*
  Sample $x_i \sim \mathcal{N}(\mu, \Sigma)$ for each $i \in \{1, \dots, N\}$;
  ▷ *Generate and repair layouts, and emulate them*
  for $i \in \{1, \dots, N\}$ **do**
    Generate layout l_i from NCA parameters x_i;
    $l_i' \leftarrow Repair(l_i)$;
    $\boldsymbol{f}_i, \boldsymbol{m}_i \leftarrow Emulate(x_i, l_i')$;
    $c_i \leftarrow Cell(\boldsymbol{m}_i)$
  ▷ *Compute improvements against the archive before insertion*
  $\mathcal{F}_c \leftarrow \{\boldsymbol{f} : (x, \boldsymbol{f}, \boldsymbol{m}) \in \mathcal{A}_c\}$ for each $c \in \mathcal{M}$ and $\mathcal{F} \leftarrow \bigcup_{c \in \mathcal{M}} \mathcal{F}_c$;
  for $i \in \{1, \dots, N\}$ **do**
    $\Delta_i^{\text{cell}} \leftarrow HV(\mathcal{F}_c \cup \{\boldsymbol{f}_i\}) - HV(\mathcal{F}_c)$;
    $\Delta_i^{\text{global}} \leftarrow HV(\mathcal{F} \cup \{\boldsymbol{f}_i\}) - HV(\mathcal{F})$;
    $\Delta_i \leftarrow \Delta_i^{\text{cell}} + \lambda \Delta_i^{\text{global}}$
  ▷ *Rank offspring and update CMA-ES*
  Sort $\{x_i\}_{i=1}^N$ by Δ_i in descending order;
  Update (μ, Σ) with the offspring;
  ▷ *Update each cell by NSGA-II survivor selection*
  for $c \in \mathcal{M}$ **do**
    $\mathcal{C}_c \leftarrow \mathcal{A}_c \cup \{(x_i, \boldsymbol{f}_i, \boldsymbol{m}_i) : c_i = c\}$;
    Update $\mathcal{A}_c$ and $survived_i$ by performing NSGA-II survivor selection on $\mathcal{C}_c$ with capacity K
  ▷ *Restart if needed*
  if $\neg survived_i$ for all offspring **then**
    Restart CMA-ES with μ randomly sampled from the global PF of $\mathcal{A}$
  $t \leftarrow t + 1$
return MOQD archive $\mathcal{A}$ of NCA generator parameters

For experiments with only layout entropy H_ℓ as a diversity measure, the result archive has 50 cells of equal-width over $H_\ell \in [0, 1]$. Within each occupied cell, we split the num of shelves range from 200 to 672 into five equal-width bands, and we place the solutions into the bands based on their number of shelves. We select one solution uniformly at random from each nonempty cell–band group. Thus, the selected set contains at most $50 \times 5 = 250$ generators.

For experiments with only the number of shelves N_s as diversity measure, the archive already has 50 equal-width cells over $[200, 672]$, so no coarsening is needed. We select up to 5 generators uniformly without replacement from each cell.

For the 2D (N_s, H_ℓ) measures, the archive uses 25 num of shelves bins over $[200, 672]$ and 10 entropy bins over $[0, 1]$. We select one generator uniformly at random from each nonempty cell, yielding $25 \times 10 = 250$ generators to be evaluated in the selected set.

The remaining real-evaluation budget caps the final batch. Failed attempts count toward the budget but are excluded from surrogate training; successful evaluations update the training dataset and ground-truth result archive.

A.3 MILP: Warehouse Domain Specific Constraints

In Section 4.3 we discussed validity and biconnectivity constraints. Here we discuss warehouse domain-specific constraints of the MILP solver.

Algorithm 2 Human-Designed Warehouse Layout Generation

Input: storage dimensions H, W; shelf-block sizes $\mathcal{B}_h, \mathcal{B}_w$; aisle sizes $\mathcal{A}_h, \mathcal{A}_w$; output directory D
Output: collection L of repaired layouts, also saved in D

$L \leftarrow \emptyset$;
$\mathcal{P} \leftarrow \mathcal{B}_w \times \mathcal{B}_h \times \mathcal{A}_w \times \mathcal{A}_h$;
for each $(b_w, b_h, a_w, a_h) \in \mathcal{P}$ **do**
  if $b_w \geq 3$ and $b_h \geq 3$ **then**
    continue;
  Initialize l as an $H \times W$ grid of empty space tiles;
  for $r \in \{0, b_h + a_h, 2(b_h + a_h), \ldots\} \cap [0, H)$ **do**
    for $c \in \{0, b_w + a_w, 2(b_w + a_w), \ldots\} \cap [0, W)$ **do**
      Fill $l[r : \min(r + b_h, H),\ c : \min(c + b_w, W)]$ with shelves;
  $l \leftarrow$ `PlaceEndpoints`(l);
  $l \leftarrow$ `RemoveIsolatedEndpoints`(l);
  $l \leftarrow$ `RemoveIsolatedShelves`(l);
  $l \leftarrow$ `AddWorkstationBlocks`(l);
  $s \leftarrow$ number of shelves in l;
  $l' \leftarrow \mathit{Repair}(l, s)$;
  if $l' \neq \perp$ **then**
    $L \leftarrow L \cup \{l'\}$;
return L;

Non-Storage Area Constraints. We require that the non-storage area does not change. We also constrain the number of workstations to be the same after repairing so that MILP does not put workstations in the storage area. Let $W \subset V$ be the vertices in the non-storage area, then:

$$\forall v \in W,\ x \in O, x_v = x_v^0. \quad \sum_{v \in V} w_v = \sum_{v \in V} w_v^0. \tag{9}$$

Number of Shelves Constraints. According to Definition 5, we require the number of shelves in the optimized layout to be in the range $[N_s^-, N_s^+]$. Therefore, we require:

$$N_s^- \leq \sum_{v \in V} s_v \leq N_s^+. \tag{10}$$

B Additional Details of Experiment Setup

B.1 Computational Resources

Since our experiments requires massive parallelization to run the MILP solvers and the LMAPF simulations, we use the following computational resources: (1) a local machine with a 64-core AMD Ryzen Threadripper 3990X CPU, 192 GB of RAM and a RTX 3090 GPU, (2) a local machine with a 64-core AMD Ryzen Threadripper 3990X CPU, 128 GB of RAM and a RTX 5090 GPU, (3) a local machine with a 64-core AMD Ryzen Threadripper 7980X CPU, 256 GB of RAM and a GTX 1080Ti GPU, (4) a local machine with a 16-core AMD Ryzen 9950X CPU, 64 GB of RAM and a RTX 4070 Super GPU, (5) a local machine with a 16-core AMD Ryzen 5950X CPU and a RTX 3080 GPU, (6) a local machine with two 48-core AMD EPYC 7642 CPUs, 256 GB of RAM, (7) a local machine with two 18-core Intel Xeon 8124M CPUs, 256 GB of RAM, (8) an HPC with numerous 64-core AMD EPYC 7742 CPUs, each with 256 GB of RAM and V100 GPUs, (9) an HPC with numerous 64-core Intel Xeon Platinum 8562Y+ CPUs, each with 950 GB of RAM and l40s GPUs, and (10) an HPC with numerous Intel Xeon Gold 6448H CPUs, each with 256 GB of RAM.

B.2 Greedily-generated Layouts

Algorithm 2 presents the procedure of greedily generating a collection of layouts by systematically varying rectangular shelf blocks and the aisles separating them. For each eligible design, it

constructs the storage area, places endpoints next to shelves, removes isolated tiles, and adds workstations. It then repairs the layout while preserving its shelf count, adds successful results to the collection L, and saves them.

The inputs H and W specify the height and width of the storage area, excluding workstation blocks. The sets $\mathcal{B}_h$ and $\mathcal{B}_w$ contain candidate shelf-block heights and widths, while $\mathcal{A}_h$ and $\mathcal{A}_w$ contain the vertical and horizontal aisle widths. The directory D stores the generated layouts. The implementation uses $H = 33$, $W = 32$, $\mathcal{B}_h = \mathcal{B}_w = \{1, \ldots, 32\}$, $\mathcal{A}_h = \{1, \ldots, 5\}$, and $\mathcal{A}_w = \{1, \ldots, 4\}$.

Line 1 initializes the empty output collection L. Line 2 forms the Cartesian product of the parameter sets, and Line 3 iterates over its designs. Lines 4 to 5 skip designs whose shelf blocks are at least 3 tiles wide and 3 tiles high, because a 3×3 block of shelves could leave the shelf in the middle not reachable. Line 6 initializes an empty storage grid. Lines 7 and 8 traverse block origins with vertical spacing $b_h + a_h$ and horizontal spacing $b_w + a_w$. Line 9 fills each block with shelves, clipping it at the grid boundary; indices are zero-based and slices exclude their upper bounds. Lines 9 to 9 close these placement loops. Line 10 calls `PlaceEndpoints`, which converts empty space tiles adjacent to shelves into endpoints. Line 11 calls `RemoveIsolatedEndpoints` to replace endpoints having no adjacent shelf with empty space. Line 12 then calls `RemoveIsolatedShelves` to replace shelves having no adjacent endpoint with empty space. Here, adjacency means in-bounds, four-neighbor adjacency. Line 13 calls `AddWorkstationBlocks` to append the non-storage area with workstations. Line 14 counts the remaining shelves. Line 15 calls Repair, which uses MILP to repair the generated layout, enforcing validity or biconnectivity constraints. Line 16 checks if the MILP repair is successful. If successful, Line 17 adds l' to L. Line 18 returns the successfully repaired set of greedily generated layouts L.

B.3 Surrogate Model Training and Exploiting

MO-DSAGE and MO-DSAGE-HVI use the same three-stage surrogate, based on the surrogate model of Zhang et al. (2023b). The first stage uses a *repair network* to predict the MILP-repaired layout given the generated unrepaired layout. The second stage uses a *occupancy network* to predict the occupancy of the robots at each vertex of the layout. The third stage uses a *final predictor* to predict the objectives and diversity measures.

Model Architecture. We one-hot encode each layout with four tile-type channels and pad it with empty tiles to 64×64. The repair and occupancy networks each use a 3×3 convolution, two 64-channel residual blocks, and a 1×1 output convolution. The repair network outputs four tile-type scores per tile. The occupancy network takes these scores as input and outputs a single occupancy channel.

The final predictor takes the predicted repaired layout, obtained by selecting and one-hot encoding the highest-scoring tile type, together with the occupancy channel. It uses four stride-2, 4×4 convolutions with 64, 128, 256, and 512 channels. Each convolution has a Leaky-ReLU activation, and the last three also use batch normalization. Separate linear heads predict the objectives and diversity measures.

Model Training. We train the three networks one at a time, with the other two networks frozen. We use a loss suited to each network: tile-wise cross-entropy for the repair network, KL divergence for the occupancy network, and mean-squared error (MSE) for the final predictor. The final predictor has one scalar output per objective and one per measure. Its loss is the mean of the per-output MSEs, with the layout-entropy MSE multiplied by 10 before averaging. Each network is trained for 20 epochs with shuffled minibatches of 64. We use a shared Adam optimizer (Kingma & Ba, 2014) with learning rate 10^{-3}, $(\beta_1, \beta_2) = (0.9, 0.999)$, and no weight decay. The model weights and optimizer state carry over between updates.

Model Exploitation Settings. We run MO-CMA-MAE in the model exploitation phase while using the surrogate model to predict the objectives and measures. Each MO-DSAGE model-exploitation phase initializes five MO-CMA-MAE emitters with zero means, step size 0.2, and 16 offspring per emitter. We run 5,000 inner iterations, giving 400,000 surrogate queries per phase. The optimization archive uses learning rate 0.1 and discount-factor search tolerance 10^{-3}; the result archive uses learning rate 1 and a capacity of 50 solutions per cell. For this search, predicted shelf counts are clipped to $[200, 672]$ and rescaled to $[0, 100]$; throughput scores are clipped to $[0, 100]$ and en-

Size	Storage area	Full layout	NCA updates	MILP limit per stage
S_2	69×65	69×69	200	2,400 s
S_3	101×98	101×102	300	5,400 s

Table 3: Scaling settings for learned generators at sizes S_2 and S_3.

tropy to $[0, 1]$. MO-DSAGE-HVI instead uses the NSGA-II cell-selection procedure described in Appendix A.1. Both methods use 50 NCA updates per layout and the downsampling protocol in Appendix A.2.

B.4 Scaling To Larger Layouts

Generation and Repair. We directly use the optimized NCA generators on the PFs to generate layouts of larger sizes. The original storage area is $A_0 = 33 \times 32 = 1{,}056$ tiles. For a target storage area A and source shelf count n, learned layouts use a relative storage-density tolerance of 10%:

$$N_s^- = \left\lceil 0.9n\frac{A}{A_0} \right\rceil, \qquad N_s^+ = \left\lfloor 1.1n\frac{A}{A_0} \right\rfloor. \tag{11}$$

We limit the number of shelves to be within a predicted range based on the size and density of the layouts so that the repaired large layouts have more chance to exhibit the same pattern as the small ones. After NCA generation, we append the fixed workstation regions and perform MILP repair for validity and biconnectivity. Table 3 lists the time limit of the MILP solver.

Greedily-Generated Baseline. In addition to optimized layouts, we also use Algorithm 2 to greedily generate larger layouts and then repair them for validity and biconnectivity while preserving the number of shelves. We evaluate every repaired layout with PIBT for 1,000 timesteps and $R = \lfloor 0.33(|V| - N_s) \rfloor$ robots.

C Additional Experimental Results

C.1 Warehouse (even)

In Section 5, we optimize the layout of the storage area and put workstations only on the left side of it to form a complete warehouse layout. Since there are much less workstations than endpoints, traffic usually concentrate the left side of the warehouse around the workstations. In this section, we experiment a different and easier warehouse setup, where we put workstations on both sides of the storage area. We refer this setup as *warehouse (even)* and the one in Section 5 as *warehouse(uneven)*. Warehouse (even) is arguably *easier* than warehouse (uneven) because we pose more workstations with the former than latter, creating more balanced traffic in the warehouse. In fact, Zhang et al. (2023a) has shown that it is much easier to optimize a high-throughput layout in warehouse (even) than in warehouse (uneven) in single-objective settings.

Figures 5 and 6 show the even-domain counterparts of Figures 3 and 4. Table 4 reports hypervolume at the size of $S_1 = 33 \times 36$. Table 5 reports the HV after scaling the optimized NCA to larger layouts. Interestingly, the greedily-generated baseline layouts is very strong across different sizes of layouts. MO-DSAGE achieves better hypervolume only in RHCR in a small size of S. In S_2 and S_3, greedily-generated layouts remain having the highest hypervolume. On the other hand, MO-DSAGE generally achieves better hypervolume than MO-CMA-MAE and CMA-MOA across different layout sizes and LMAPF algorithms.

The advantage of greedily-generated layouts in warehouse (even) comes from the fact that it is inspired and designed specifically for this setup. Since workstations are added to both sides of the layout and the goal locations of the robots are sampled uniformly at random, the traffic of the robots is balanced much more evenly than warehouse (uneven). Meanwhile, greedily-generated layouts are generated by uniformly placing the shelves in different ways in the warehouse. Therefore, it is easy for some of the such generated layouts to demonstrate high throughput and high storage capacity at the same time.

PIBT				RHCR			
Connected		Biconnected		Connected		Biconnected	
Algorithm	HV ↑	Algorithm	HV ↑	Algorithm	HV ↑	Algorithm	HV ↑
MO-DSAGE	2041.2 ± 32.5	MO-DSAGE	2155.7 ± 16.0	MO-DSAGE	1951.3 ± 73.7	**MO-DSAGE**	$\mathbf{2171.6 \pm 108.2}$
MO-CMA-MAE	1890.5 ± 86.7	MO-CMA-MAE	2035.2 ± 25.2	MO-CMA-MAE	1756.0 ± 110.7	MO-CMA-MAE	2020.4 ± 31.8
CMA-MOA	1917.9 ± 26.5	CMA-MOA	2062.1 ± 21.5	CMA-MOA	1713.4 ± 112.2	CMA-MOA	1872.7 ± 49.4
Greedy	2132.5	**Greedy**	**2166.5**	Greedy	2169.5	Greedy	2087.8

Table 4: Hypervolumes (HVs) of our methods comparing with the baseline with both PIBT and RHCR in warehouse (even). Results are reported in the form of $x \pm y$ where x is the mean and y is the 95% Student-t confidence-interval half-widths over four runs.

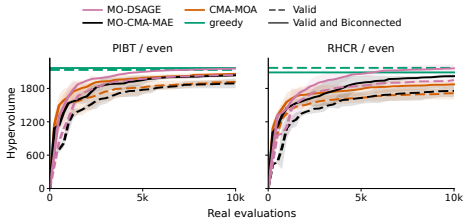


Figure 5: Hypervolume optimization curves in warehouse (even) over 10k evaluations. Shading shows 95% confidence intervals.

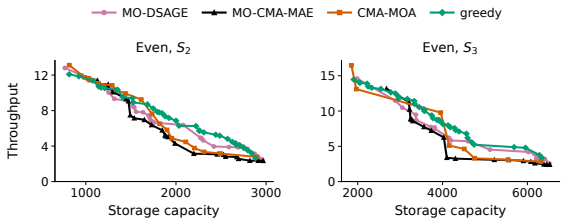


Figure 6: Scaling warehouse (even) to S_2 and S_3 with PIBT and biconnected repair. NCA updates and simulation counts match Figure 4.

Size	MO-DSAGE	MO-CMA-MAE	CMA-MOA	Greedy
S_2	14,245.0	12,499.8	13,393.3	**15,290.0**
S_3	32,559.5	29,877.5	28,754.3	**37,413.3**

Table 5: HVs of scaled layouts for warehouse (even).

However, we argue that the advantage of the greedily-generated layouts do not automatically translate to other settings. As shown in Section 5, for example, they demonstrate much worse hypervolume than other search-based methods in warehouse (uneven), when we change the workstation layouts. In real-world warehouses, the distribution of the goal locations of the robots is much more complicated and uneven than uniformly sampling from all possible goals. These will be the scenarios where our proposed MOLO methods becomes useful in terms of searching for high-quality layouts.

Longer-Budget MO-DSAGE-HVI Comparison. Another possible cause of greedily-generated layouts being better than search-based methods is that the latter methods are not converged within 10k evaluations. Therefore, we run conduct a comparison between MO-DSAGE-HVI and the greedily-generated baseline with PIBT in warehouse (even) domain, while running MO-DSAGE-HVI for a much higher number of $N_{\text{eval}} = 30,000$ evaluations. We use the number of shelves as the diversity measure and the same objectives presented in Section 4.1. We set $\lambda = 50$.

Figure 7 shows the hypervolume optimization curve over 30k evaluations. MO-DSAGE-HVI reaches HV 2208.2 ± 8.7, compared to 2166.5 for the greedy baseline, a 1.9% improvement in mean HV; all four optimization runs exceed the Greedy reference. The interval is a 95% Student-t confidence interval across optimization runs, treating the evaluated Greedy front as fixed. As discussed above, we conjecture that the relevative small degree of improvement comes from the fact that the greedily-generated layouts are already very high-quality. Nevertheless, our search-based MOLO algorithm is able to replicate such a high-quality PF given enough number of evaluations.

C.2 Ablation with Global HVI

We first analyze whether optimizing both global and local PF using MO-DSAGE-HVI gives better global PF than MO-DSAGE. MO-DSAGE-HVI ranks offspring by $\Delta^{\text{cell}} + \lambda \Delta^{\text{global}}$ during surrogate search (Algorithm 1). We compare MO-DSAGE with MO-DSAGE-HVI at $\lambda = 50$. Figures 9a

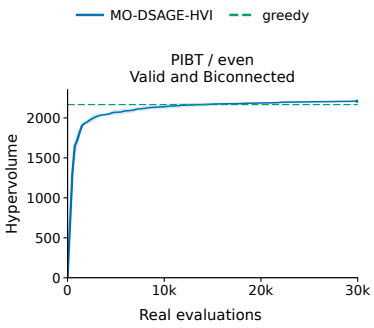


Figure 7: MO-DSAGE-HVI with num-of-shelves-only diversity versus Greedily-generated on PIBT even biconnected layouts up to 30k evaluations. The curve is a mean over four seeds; shading shows pointwise 95% Student-t confidence intervals.

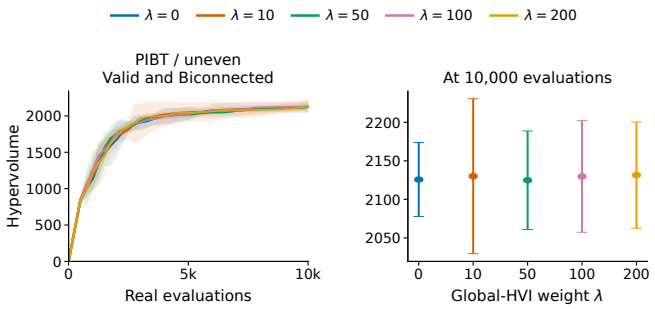


Figure 8: Global-HVI coefficient ablation for MO-DSAGE-HVI on PIBT uneven biconnected layouts with num-of-shelves-only diversity. Left: HV optimization curves with pointwise 95% Student-t confidence intervals. Right: mean HV at 10,000 real evaluations with the corresponding confidence intervals. Results for each coefficient are averaged over four runs.

λ	0	10	50	100	200
HV	2125.7 ± 48.0	2130.1 ± 100.5	2124.8 ± 63.9	2129.7 ± 72.4	2131.4 ± 68.9

Table 6: Mean HV $\pm$ 95% Student-t confidence-interval half-width at 10,000 evaluations, with four runs for every global-HVI weight.

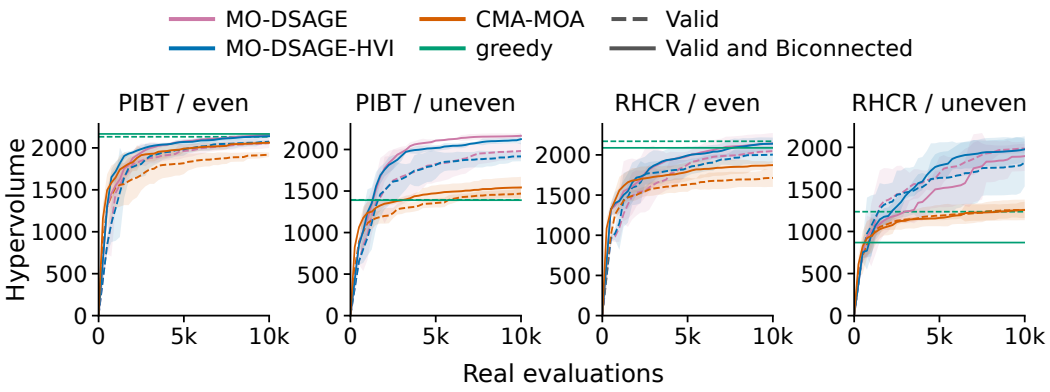
and 9b compares them across LMAPF planners (RHCR and PIBT), warehouse domains (even and uneven). Figure 9a uses the number of shelves as the diversity measure, while Figure 9b uses the layout entropy as the diversity measure. Table 7 reports the numerical results of the hypervolume. With the diversity of the number of shelves, the mean HV of MO-DSAGE-HVI differs from MO-DSAGE by approximately -8.4% to $+4.3\%$; with the layout entropy, the range is approximately -0.8% to $+6.0\%$. Thus, the global-HVI search procedure does not improve uniformly on MO-DSAGE.

We then further analyze whether different λ yields better results on PIBT, uneven, biconnected layouts with num-of-shelves-only diversity. We perform 4 runs, each with $C = 50$ and $N_{\text{eval}} = 10,000$. Figure 8 and Table 6 show very similar final means in all weights tested: the largest and smallest means differ by less than 0.4%. The uncertainty intervals are substantially larger than these differences. These results do not establish a preferred weight for MO-DSAGE-HVI.

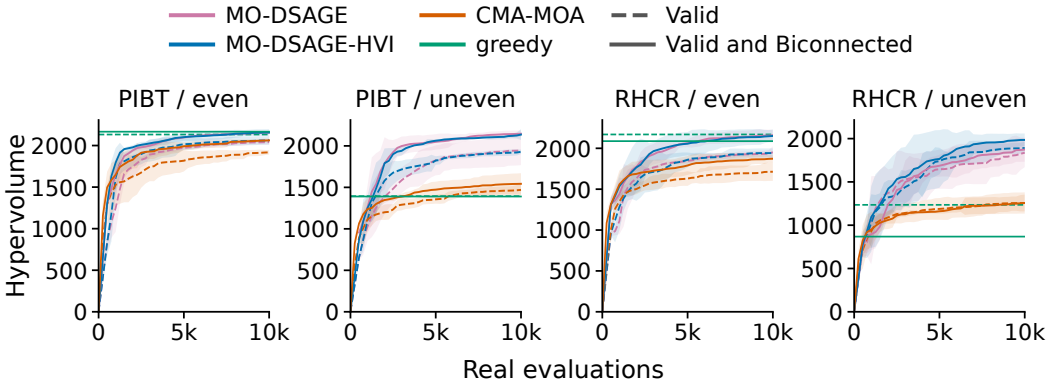
C.3 Ablation on Diversity Measures

We compare shelf count N_s alone, layout entropy H_ℓ alone, and their combination (N_s, H_ℓ). For a repaired storage area, layout entropy is computed from the frequencies p_j of the 81 possible overlapping 2×2 shelf/endpoint/empty-space patterns: $H_\ell = -\sum_{j=1}^{81} p_j \log p_j / \log 81$, with $0 \log 0 = 0$. We include a method and setting only if *all three* diversity measure choices have completed all optimization seeds at 10,000 real evaluations. This yields four comparisons: MO-CMA-MAE and MO-DSAGE-HVI, each on PIBT even and uneven biconnected layouts.

Table 8 and Figure 10 show that layout entropy has the highest observed mean HV in all four included comparisons. Adding shelf count to entropy does not increase the mean in either method or domain.



(a) Number of shelves as the diversity measure.



(b) Layout entropy as the diversity measure.

Figure 9: Optimization curves comparing MO-DSAGE-HVI with all other methods. Shading gives 95% confidence intervals.

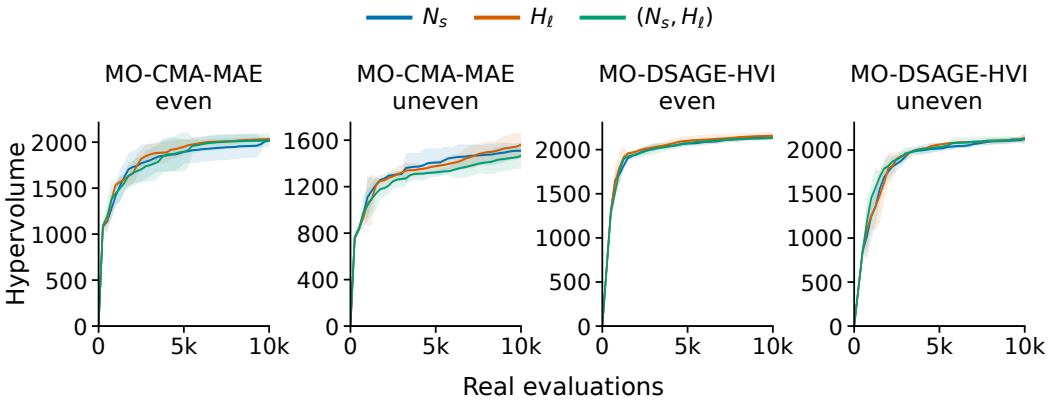


Figure 10: Optimization curves for the three-measure comparisons in. Curves are averaged over four runs and shading gives 95% confidence intervals.

C.4 Instability of NCA Scaling

We investigate why MO-DSAGE exhibits a bad scaling result in Section 5.2. Critically, we observe that the PF of MO-DSAGE in S_3 is concentrated in the region of high storage capacity. Therefore, we show the scatter plot of all layouts generated by the NCA generators on the PF in Figure 11. In-

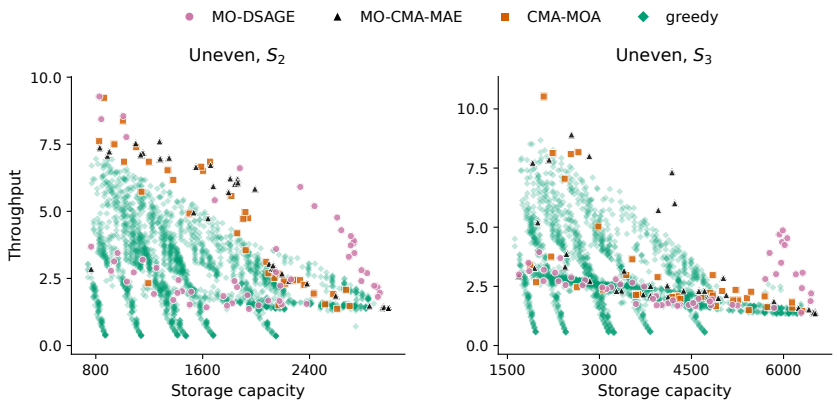
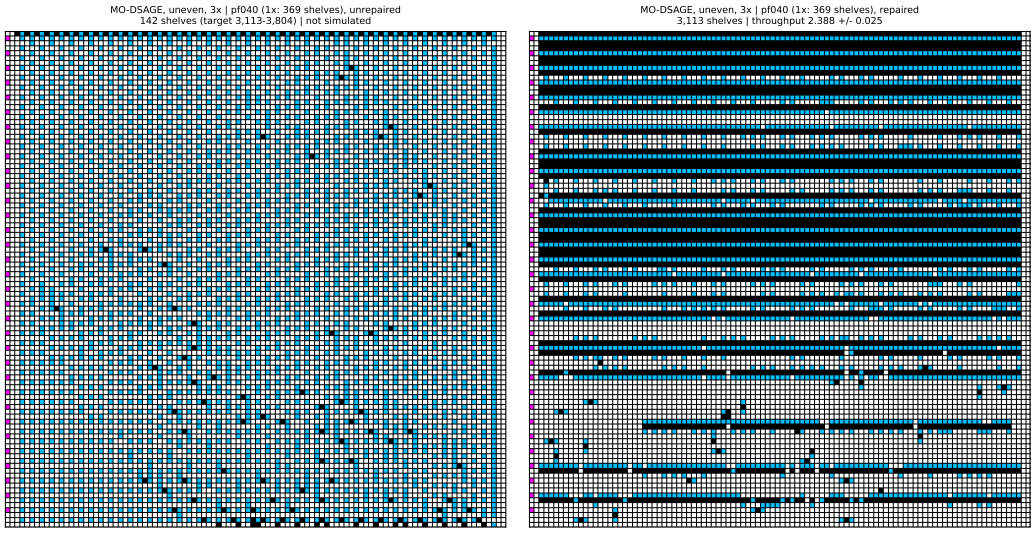


Figure 11: Scatter plots of all layouts generated by the NCA generators on the PF optimized with PIBT and warehouse (uneven).



(a) Unrepaired (b) Repaired

Figure 12: Example unrepaired and repaired layouts for PIBT in warehouse (uneven) at the size of S_3.

Setting	MO-DSAGE	MO-DSAGE-HVI	CMA-MOA	Greedy
Num-of-shelves-only diversity				
PIBT / even / C	2057.2 ± 58.6	2073.7 ± 23.5	1917.9 ± 26.5	2132.5
PIBT / even / B	2156.5 ± 21.2	2139.3 ± 25.3	2062.1 ± 21.5	2166.5
PIBT / uneven / C	1982.1 ± 95.8	1916.9 ± 59.0	1469.5 ± 71.6	1394.0
PIBT / uneven / B	2164.8 ± 34.0	2124.8 ± 63.9	1542.8 ± 128.2	1389.8
RHCR / even / C	2047.7 ± 11.5	2003.6 ± 106.1	1713.4 ± 112.2	2169.5
RHCR / even / B	2142.1 ± 130.5	2139.4 ± 76.2	1872.7 ± 49.4	2087.8
RHCR / uneven / C	1985.5 ± 55.7	1819.4 ± 282.7	1256.6 ± 122.9	1235.1
RHCR / uneven / B	1895.9 ± 176.1	1976.5 ± 153.8	1256.1 ± 96.9	868.4
Layout-entropy-only diversity				
PIBT / even / C	2041.2 ± 32.5	2062.5 ± 51.3	1917.9 ± 26.5	2132.5
PIBT / even / B	2155.7 ± 16.0	2157.0 ± 40.0	2062.1 ± 21.5	2166.5
PIBT / uneven / C	1944.5 ± 176.7	1928.4 ± 27.2	1469.5 ± 71.6	1394.0
PIBT / uneven / B	2146.6 ± 38.9	2136.2 ± 65.5	1542.8 ± 128.2	1389.8
RHCR / even / C	1951.3 ± 73.7	1941.4 ± 57.9	1713.4 ± 112.2	2169.5
RHCR / even / B	2160.5 ± 66.6	2151.3 ± 78.8	1872.7 ± 49.4	2087.8
RHCR / uneven / C	1837.2 ± 166.7	1891.7 ± 70.6	1256.6 ± 122.9	1235.1
RHCR / uneven / B	1873.3 ± 140.4	1984.9 ± 74.8	1256.1 ± 96.9	868.4
Combined num-of-shelves and layout-entropy diversity				
PIBT / even / B	—	2132.5 ± 18.1	2062.1 ± 21.5	2166.5
PIBT / uneven / B	—	2123.5 ± 25.7	1542.8 ± 128.2	1389.8

Table 7: HV comparing MO-DSAGE and MO-DSAGE-HVI. Results are reported in the format of $x \pm y$ where x is the mean and y is 95% Student-t confidence-interval over four runs. C denotes enforcing validity constraints only and B denotes enforcing both validity and biconnectivity constraints.

Method	Warehouse domain	N_s	H_ℓ	(N_s, H_ℓ)
MO-CMA-MAE	Even	2016.4 ± 18.4	2035.2 ± 25.2	2018.9 ± 74.6
MO-CMA-MAE	Uneven	1509.4 ± 65.7	1560.8 ± 107.5	1464.8 ± 115.9
MO-DSAGE-HVI	Even	2139.3 ± 25.3	2157.0 ± 40.0	2132.5 ± 18.1
MO-DSAGE-HVI	Uneven	2124.8 ± 63.9	2136.2 ± 65.5	2123.5 ± 25.7

Table 8: Three-way diversity-measure ablation on biconnected layouts. Every entry is mean HV $\pm$ 95% Student-t confidence-interval half-width over four seeds at 10,000 evaluations.

terestingly, layouts with low storage capacity for MO-DSAGE have low throughput. We conjecture that these layouts have low throughput because the MILP solver does not successfully repair the layouts within the given time limit. Evidence is shown in Figure 12, where a large discrepancy is shown between a repairable and unrepaired layout. Future works can investigate how to improve the optimization algorithm so that the generated layout can be directly valid and biconnected, instead of relying on MILP solver to enforce such constraints.