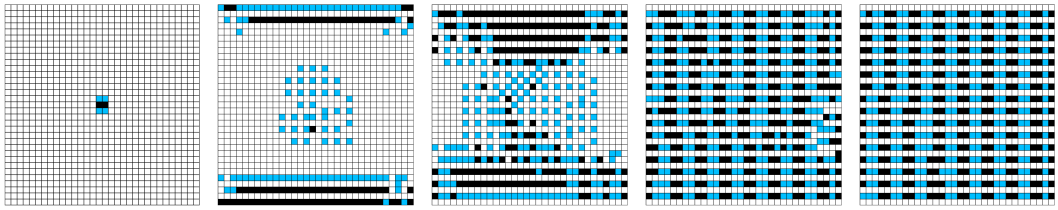
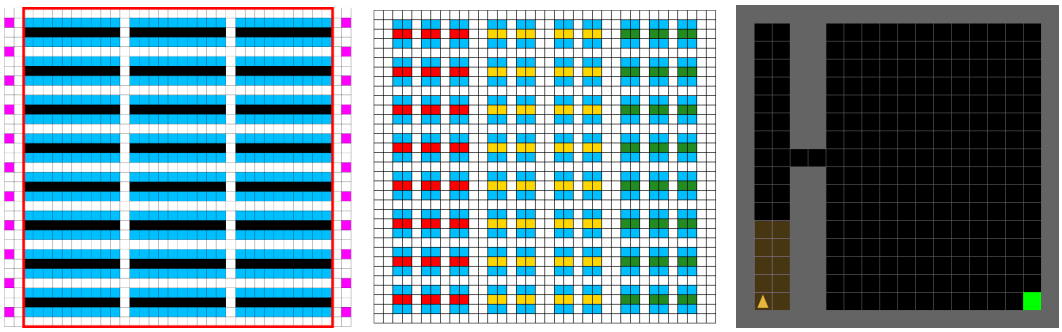


(a) Iteration 0 (b) Iteration 5 (c) Iteration 10 (d) Iteration 30 (e) Iteration 50

Figure 6: Example NCA environment generation process with 50 iterations, which starts from a fixed initial environment (Figure 6a) and iteratively generates the rest of the environments (Figures 6b to 6e).



(a) Warehouse (b) Manufacturing (c) Maze

Figure 7: Example environments of the warehouse, manufacturing, and maze domains.

A NCA Generation Process

Figure 6 shows an example NCA generation process with 50 iterations.

B Domain Details

B.1 Warehouse

Environment. We use the workstation scenario from previous work [51]. Figure 7a shows an example. A warehouse environment has four tile types. Black tiles are shelves for storing goods and also serve as obstacles. Blue tiles are endpoints where the agents park and interact with shelves. White tiles are empty spaces. Pink tiles are workstations where agents interact with human staff. Agents can traverse all non-black tiles. The goal location (= task) of each agent alternates between randomly selected workstations and endpoints. Furthermore, the NCA generator only generates the *storage area* inside the red box in Figure 7a, which contains only black, blue, and white tiles. The *non-storage area* outside the red box is kept unchanged. This is because the locations of the workstations are usually fixed along the borders of the real-world warehouses.

Domain-Specific Constraints. The constraints of the warehouse environment include: (1) all non-black tiles are connected so that the agents can reach all endpoints and workstations, (2) each black tile is adjacent to at least one blue tile and vice versa, ensuring that agents can interact with shelves via endpoints, (3) the number of black tiles is equal to a predefined number (N_s or N_{s_eval} in Table 1) so that all generated warehouse environments have the same storage capability, and (4) the non-storage area is kept unchanged. The detailed MILP formulation of the constraints is included in the previous paper [51].

Agent-based Simulator. Following previous work [51], we use Rolling-Horizon Collision Resolution (RHCR) [28], a state-of-the-art lifelong MAPF planner, in our simulations. In lifelong MAPF, agents are constantly assigned new tasks once they finish their previous ones. At every h timesteps, RHCR plans collision-free paths for all agents for the next w timesteps ($w \geq h$). Following the design recommendation from the previous work [28], we use $w = 10$ and $h = 5$. We set the initial locations of the agents uniformly at random from the non-shelf tiles. We use two different task assigners,

leading to two variations of the warehouse domain. First, we adopt the task assigner from the previous work [51] that assigns the next task (workstation or endpoint) uniformly at random. We refer to this variant as *warehouse (even)*. Second, we extend the previous task assigner by asking it to select the next workstation with uneven probabilities. Specifically, the workstations on the left border are 5 times more likely to be selected than those on the right. The endpoints remain being evenly selected. We refer to this variant as *warehouse (uneven)*, which is more challenging than warehouse (even) because the more frequently visited left border workstations could make the simulation to be congested.

B.2 Manufacturing

Environment. Figure 7b shows an example of a manufacturing environment. Each manufacturing environment has five tile types. The blue (endpoint) and white (empty space) tiles are the same as the warehouse environments. The red, green, and yellow tiles represent three types of workstations. Agents can traverse only blue and white tiles. The task of each agent is to go to an endpoint adjacent to a red, green, and yellow workstation in order and stay there for t_r, t_g, and t_y timesteps, respectively. The NCA generator generates the entire environment. Since the manufacturing domain has multiple types of workstations, one of the challenges of generating high-performance manufacturing environments is to find a reasonable ratio of the number of different workstations.

Domain-Specific Constraints. In line with the warehouse domain, we constrain the manufacturing environments such that (1) all blue and white tiles are connected so that the robots can reach all endpoints, (2) each blue tile is adjacent to at least one of red, green, or yellow tiles and vice versa so that the robots can interact with the workstations via endpoints, (3) there is at least one red, one green, and one yellow tile to support the cyclical nature of manufacturing tasks, where a complete cycle involves sequential visits to endpoints next to red, green, and yellow workstations.

Agent-based Simulator. The agent-based simulator is the same as the one used in the warehouse domain except that (1) we ask each agent to stay for $t_r = 2$, $t_g = 5$, and $t_y = 10$ timesteps after arriving at their goal endpoints next to red, green, and yellow workstations, respectively, and (2) the task assigner assigns the next task to an endpoint, chosen uniformly at random, near the red, green, and yellow workstations in order.

B.3 Maze

Environment. We use the maze domain from previous works [1, 5, 9, 37]. Figure 7c shows an example maze environment. A maze environment has two tile types: wall (gray) and empty space (black). The yellow triangle represents the agent, and the green square is the goal location. We omit the agent and the goal location when generating the environments.

Domain-Specific Constraints. We do not have any domain-specific constraints for the maze domain.

Agent-based Simulator. We use a trained ACCEL agent [9]. The observation space is a 5×5 grid in front of it. The agent can turn left or right in its current tile, move forward to the adjacent tile, or stay in the current tile. We assign the start and goal locations of the agent to the first and last tile of the longest shortest path in the environment. We limit the time horizon to 2 times the number of tiles in the environment, which is 648 for environments of size S, and 8,712 for environments of size S_{eval}, respectively. We stop the simulation early if the agent reaches the goal.

C Experiment Details

C.1 NCA Setup

C.1.1 NCA Generator Architecture

Figure 8 shows the model architecture of the NCA generator. The generator has 3 convolutional layers of kernel size 3, stride 1 and padding 1. The first 2 convolutional layers have 32 output channels and are followed by a ReLU activation. The last layer has 3 output channels and is followed by a sigmoid function. This configuration guarantees that the width, height, and number of channels of the input and output tensors are the same.

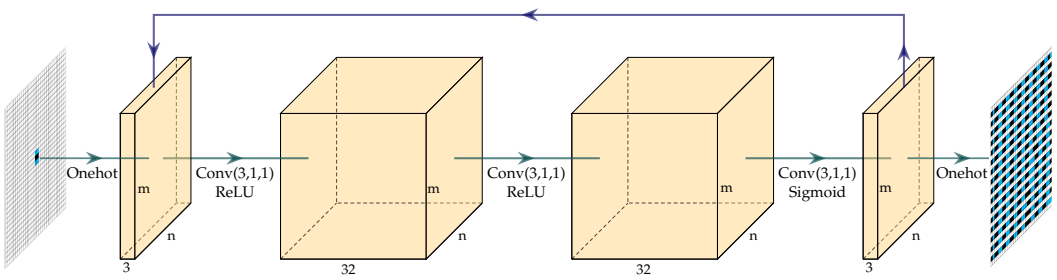
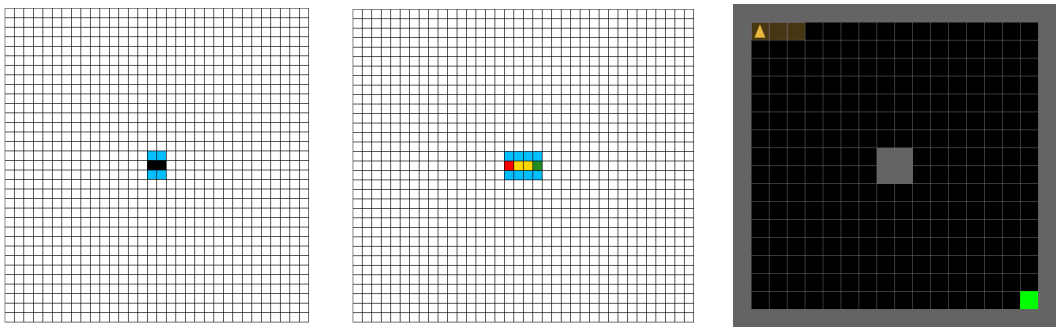


Figure 8: Architecture of the NCA generator. Starting with a fixed onehot-encoded initial environment, the generator transforms an initial environment iteratively to generate the final environment.



(a) Warehouse (b) Manufacturing (c) Maze

Figure 9: Initial environments of NCA generation.

C.1.2 Initial Environments

Figure 9 shows the initial environments of size S for all domains. They are characterized by a small central block of non-empty-spaces surrounded by empty spaces. The initial environments of size S_{eval} maintains the same central blocks and are surrounded by more expansive empty spaces.

C.1.3 Postprocessing Generated Environments

After generating an environment and before repairing it, we may need to postprocess the environment to prepare for repairing.

Warehouse. In the warehouse domain, we generate a 32×33 and 98×101 storage area and add fixed non-storage area to the left and right side to form an environment of size $S = 36 \times 33$ and $S_{eval} = 102 \times 101$, respectively.

Manufacturing. In the manufacturing domain, we directly generate environments of size $S = 36 \times 33$ and $S_{eval} = 102 \times 101$.

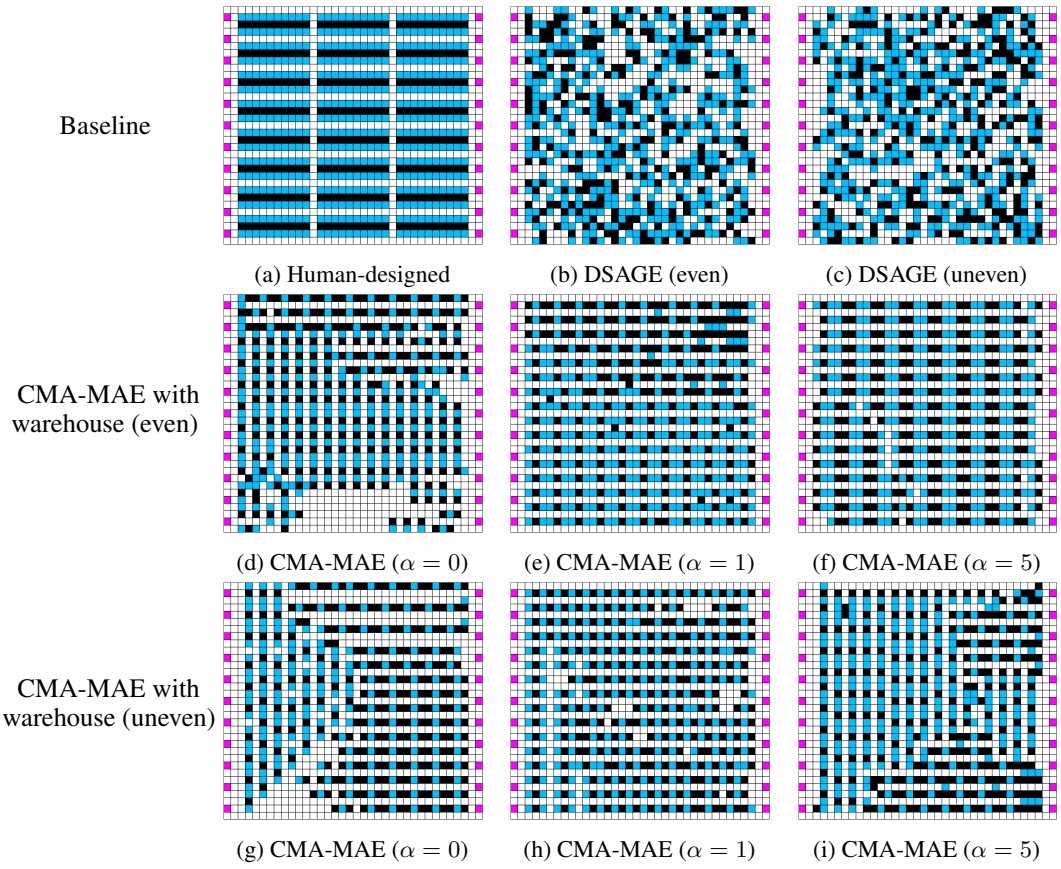
Maze. In the maze domain, we generate a 16×16 environment and surround it with walls to create a $S = 18 \times 18$ environment. Similarly, we generate a 64×64 environment and surround it with walls to create a $S_{eval} = 66 \times 66$ environment.

C.2 QD Setup

C.2.1 Objective

Warehouse. We use f_{res} and f_{opt} in Section 4 as the objective functions. We set $p_i = 1$ for all i, meaning that all tiles have the same weight while computing the similarity score.

Manufacturing. We use f_{res} and f_{opt} in Section 4 as the objective functions. We set $\alpha = 5$ as it results in the most scalable environments in the warehouse domain. Suppose $\mathbf{x}_i$ denotes the i^{th} tile of



(a) Human-designed (b) DSAGE (even) (c) DSAGE (uneven)

(d) CMA-MAE ($\alpha = 0$) (e) CMA-MAE ($\alpha = 1$) (f) CMA-MAE ($\alpha = 5$)

(g) CMA-MAE ($\alpha = 0$) (h) CMA-MAE ($\alpha = 1$) (i) CMA-MAE ($\alpha = 5$)

Figure 10: Baseline and NCA-generated warehouse environments of size S.

the environment $\mathbf{x} \in \mathbf{X}$. For the objective functions, we let $p_i = 5$ if $(\mathbf{x}_{in})_i$ is a workstation, and $p_i = 1$ otherwise. Intuitively, we give more rewards to the environments in which workstations are not changed during MILP repair.

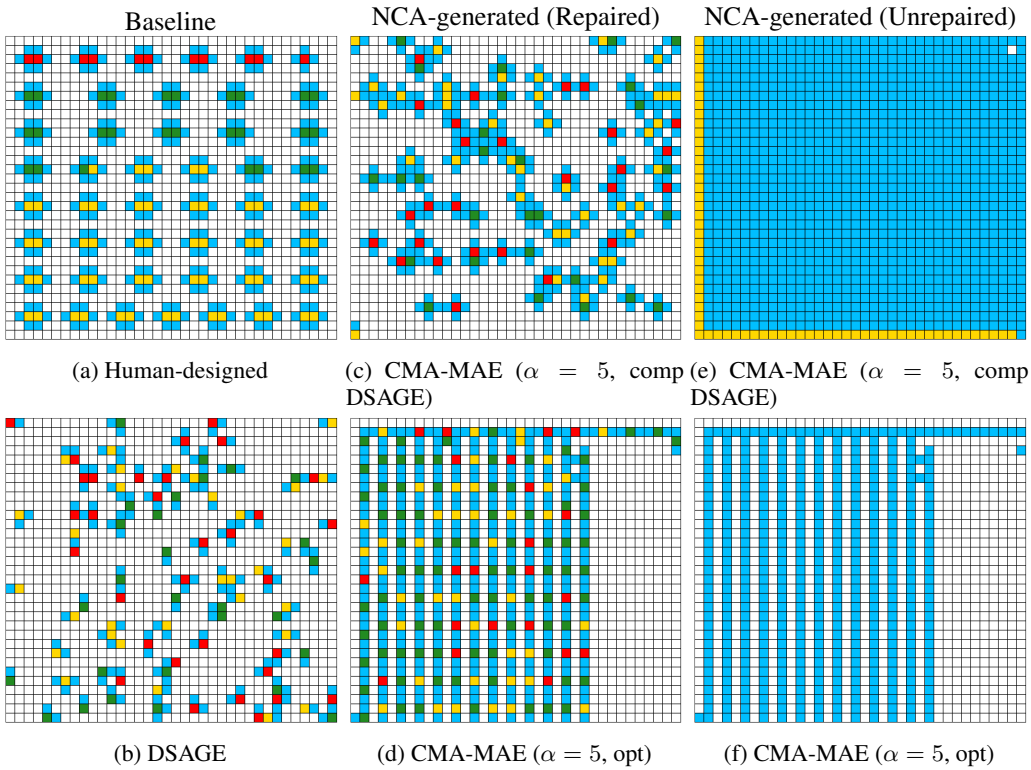
Maze. We use the same binary objective from the previous work [1].

C.2.2 Archive Dimensions

Warehouse. We use a 100×100 archive for the warehouse domain. We set the range of the connected shelf components to be $[140, 240]$, following previous work [51], and the range of the environment entropy to be $[0, 1]$. In DSAGE, we follow previous work to downsample the archive to get a subset of elites [1, 51]. We set the downsampled archive dimension to be $[50, 25]$, with 50 on the dimension of the number of connected shelf components and 25 on the environment entropy dimension. We use an irregular dimension because DSAGE fails to diversify the environment entropy measure (introduced in Appendix E.5) and we need to use a higher downsampling resolution on the number of connected shelf components to sample a reasonable number of elites.

Manufacturing. We use a 100×100 archive for the manufacturing domain. We set the range of the number of workstations to be $[0, 600]$ to allow CMA-MAE to explore a wide range of environments with different number of workstations, and the range of the environment entropy to be $[0, 1]$. We use an irregular 25×10 downsampled archive with higher resolution on the number of workstations for the same reason as the warehouse domain

Maze. Following previous work [1], we use a 256×162 archive. We set the range of the number of walls to be $[0, 256]$, where 256 is the total number of tiles in the environments of size S (excluding the appended walls around the border), and the range of average agent path length to be $[0, 648]$, where 648 is the time horizon of the agent-based simulation in environments of size S.



(a) Human-designed

(c) CMA-MAE ($\alpha = 5$, comp DSAGE)

(e) CMA-MAE ($\alpha = 5$, comp DSAGE)

(b) DSAGE

(d) CMA-MAE ($\alpha = 5$, opt)

(f) CMA-MAE ($\alpha = 5$, opt)

Figure 11: Baseline and NCA-generated manufacturing environments of size S.

C.2.3 CMA-MAE

For the warehouse and manufacturing domains, we use an archive learning rate of 0.01, while, for the maze domain, we use 0.5. We use a higher archive learning rate for the maze domain because a higher learning rate promotes CMA-MAE to explore the measure space. Since the binary objective function of the maze domain (introduced in Section 5) can be trivially optimized, we promote CMA-MAE to do more exploration than exploitation in the maze domain. In all domains, the initial batch of solutions is sampled from a multivariate Gaussian distribution with a mean of 0 and a standard deviation of 0.2.

C.3 Compute Resource

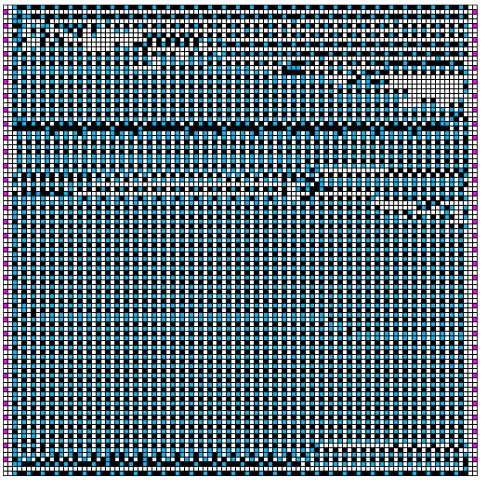
We run our experiments on 4 local machines and 2 high performing clusters, namely (1) a local machine with a 16-core AMD Ryzen 9 5950X CPU, 32 GB of RAM, and a Nvidia RTX 3080 GPU, (2) a local machine with a 64-core AMD Ryzen Threadripper 3990X, 192 GB of RAM, and an Nvidia RTX 3090Ti GPU, (3) a local machine with a 64-core AMD Ryzen Threadripper 3990X, 64 GB of RAM, and an Nvidia RTX A6000 GPU, (4) a local machine with a 64-core AMD Ryzen Threadripper 3990X, 64 GB of RAM, and an Nvidia RTX 3090 GPU, (5) a high-performing clusters with a V100 GPU, 200 Xeon CPUs each with 4 GB of RAM, and numerous 32-core AMD EPYC 7513 CPUs, each with up to 248 GB of RAM, (6) a high-performing cluster with a V100 GPU and heterogeneous CPUs. We perform our experiments in different machines because our experiments are not sensitive to runtime and require massive parallelization. We measure all CPU runtimes in machine (1).

C.4 Implementation

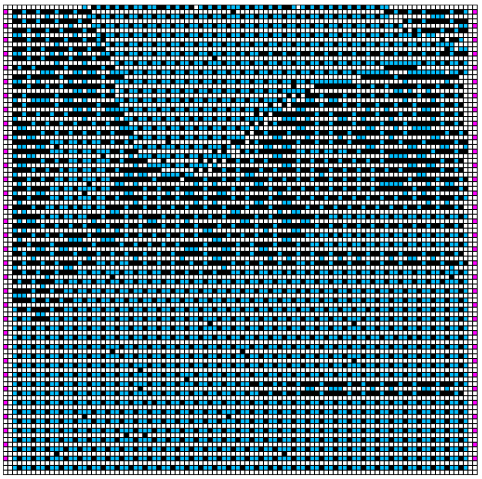
We implement CMA-MAE with Pyribs [45], the NCA generators in PyTorch [38], and the MILP solver with IBM's CPLEX library [21] in Python.

Compute Constraint of MILP. In the CPLEX MILP solver, we constrain the maximum number of threads to 1 and 28 for environments of size S and S_{eval}, respectively. We set the deterministic

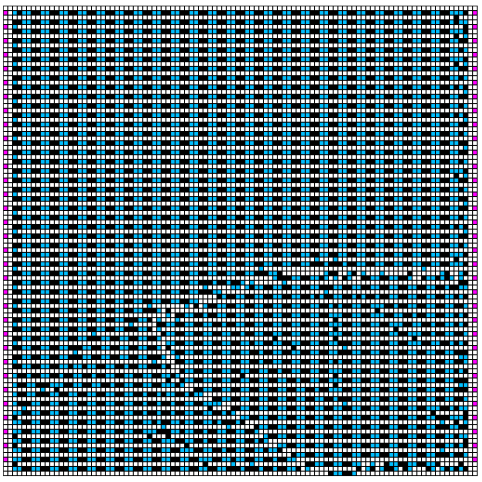
Warehouse (even)



(a) CMA-MAE ($\alpha = 0$)

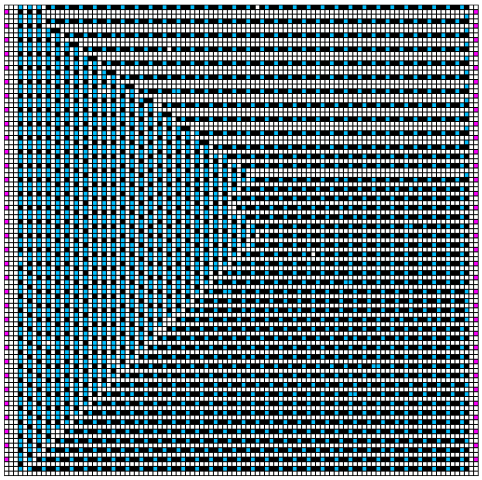


(b) CMA-MAE ($\alpha = 1$)

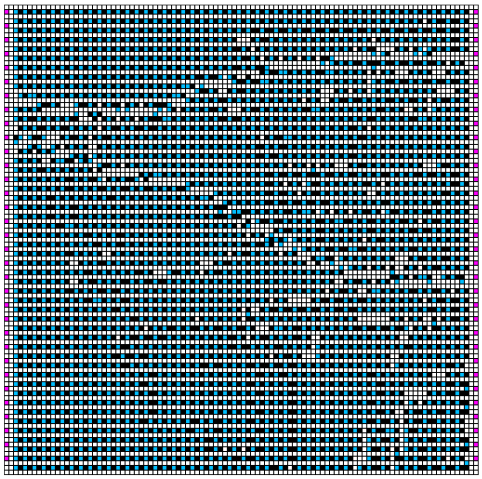


(c) CMA-MAE ($\alpha = 5$)

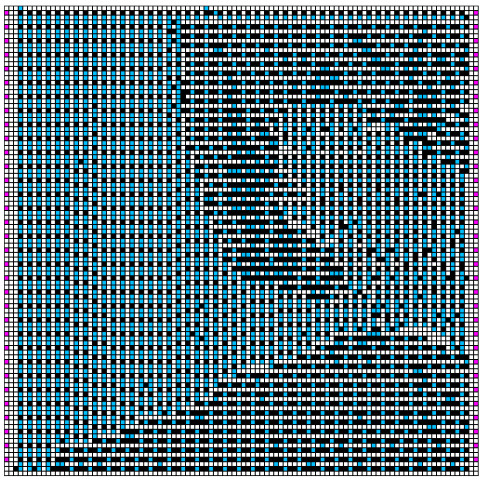
Warehouse (uneven)



(d) CMA-MAE ($\alpha = 0$)

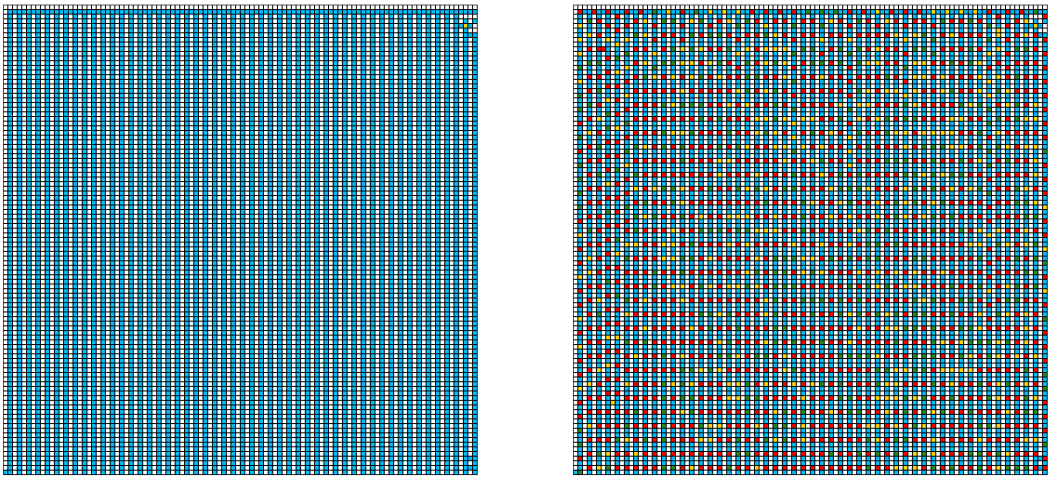


(e) CMA-MAE ($\alpha = 1$)



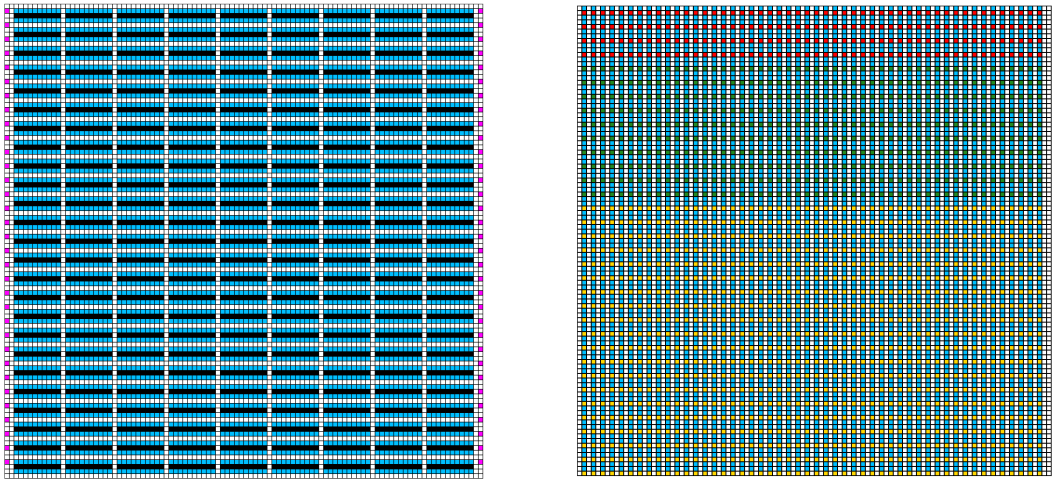
(f) CMA-MAE ($\alpha = 5$)

Figure 12: NCA-generated warehouse environments of size S_{eval}.



(a) Unrepaired. (b) Repaired.

Figure 13: NCA-generated manufacturing environment of size S_{eval} with CMA-MAE ($\alpha = 5$).



(a) Warehouse (b) Manufacturing

Figure 14: Human-designed warehouse and manufacturing environments of size S_{eval}.

runtime limit to be $6e4$ ticks for size S and $3.6e6$ ticks for size S_{eval} in CPLEX. The deterministic runtime limit in CPLEX is a metric that provides a consistent measure on the "effort" made by the MILP solver, measured in "ticks", a CPLEX abstract unit.

D Baseline and NCA-generated Environments

D.1 NCA-generated Environments

Warehouse. Figures 10d to 10i show the NCA-generated warehouse environments of size S. Figure 12 shows those of size S_{eval}. For the environments of warehouse (even), larger α values lead the optimal NCA generators in the result archives to generate environments with clearer patterns. Specifically, with $\alpha = 0$ (Figure 10d), the environment has a large area of empty spaces at the bottom. With $\alpha = 1$ (Figure 10e), the area of empty spaces is gone but the distribution of shelves is not even. With $\alpha = 5$ (Figure 10f), the generated environment is mostly filled with repeated blocks of 1×2 shelves and the space in between the block is filled with endpoints. The trend is similar for environments of size S_{eval} in the warehouse (even) domain (Figures 12a to 12c). With $\alpha = 5$, the generated environment possesses the most regularized pattern.

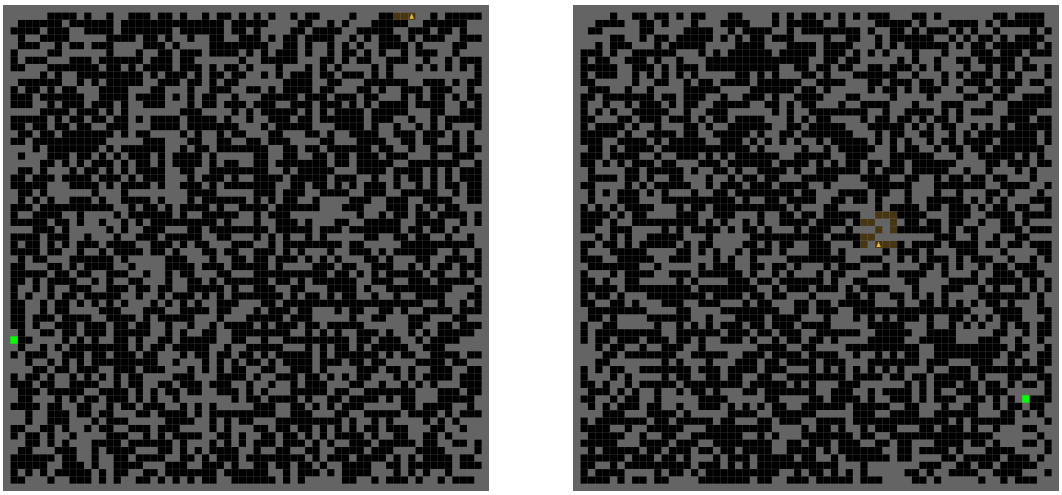


Figure 15: Baseline maze environments of size S_{eval}.

For the warehouse (uneven) domain, larger α values do not always generate clearer patterns. Specifically, with size S, the environments generated with $\alpha = 0$ (Figure 10g) and $\alpha = 5$ (Figure 10i) have similar patterns. Both environments have fewer shelves on the left part of the environments and cluster more shelves on the right so that the agents have more traversable tiles to resolve conflicts near the more frequently visited workstations on the left border. Such pattern can also be found in size S_{evals} (Figures 12d and 12f). Notably, however, with size S_{eval}, the environment with $\alpha = 5$ has more traversable tiles near the workstations on the left border than that with $\alpha = 0$, resulting in higher success rate as shown in Table 2 of Section 6.

Manufacturing. Figures 11c and 11d show the NCA-generated environments of size S. The one generated by the optimal NCA generator (Figure 11d) possesses more regularized patterns than the one used to compare with DSAGE (Figure 11c). However, in their corresponding unrepaired environments (Figures 11e and 11f), the NCA generators only generate empty spaces and endpoints and rely on the MILP solver to put the workstations in the environment. We conjecture that for the manufacturing domain, the number of endpoints is more important than the ratio between different types of workstations because more endpoints allow the agents to perform tasks in parallel around the workstations. Similarly, Figure 13 shows the unrepaired and repaired NCA-generated environments of size S_{eval}. The unrepaired environment (Figure 13a) has no workstations at all, while the repaired environment has randomly placed workstations by the MILP solver.

D.2 Baseline Environments

Warehouse. Figure 10a shows the human-designed warehouse environment of size S for both variants of warehouse domains. It is taken from the previous works [29, 28, 51], where 1×10 blocks of shelves are repeatedly placed, and each shelf is surrounded by two endpoints from the top and bottom of it. We scale this pattern to create the human-designed warehouse environment of size S_{eval}, shown in Figure 14a. Figures 10b and 10c shows the DSAGE-optimized warehouse environments of size S for warehouse (even) and warehouse (uneven), respectively. Both of them have no clear regularized patterns.

Manufacturing. Figures 11a and 14b show the human-designed manufacturing environments of size S and S_{eval}, respectively. We create the human-designed environment of size S (Figure 11a) such that (1) the number of workstations mirrors that in the DSAGE-optimized environment so that approximately the same number of agents can operate in parallel around the workstations, and (2) the ratio of different types of stations approximates $t_r : t_g : t_y = 2 : 5 : 10$ so that the environment has more workstations that require agents to stay longer.

To create these human-designed environments, we draw insights from some NCA-generated environments. Notably, the warehouse (even) environment with $\alpha = 5$ (Figure 10f) has the best scalability (shown in Section 6). This informs the design of the human-designed manufacturing environment

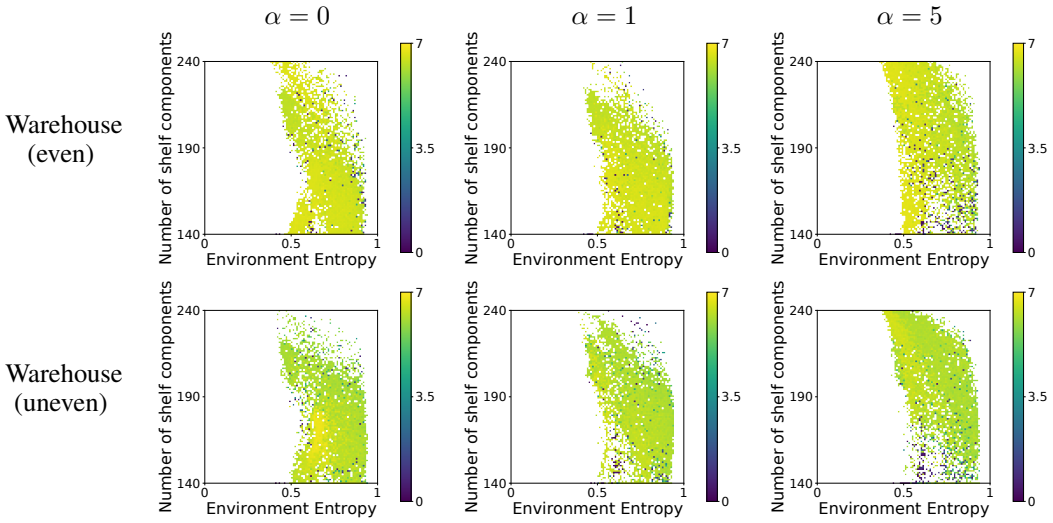


Figure 16: Example result archives of CMA-MAE + NCA in warehouse domains with different α values.

of size S, displayed in Figure 11a, where we evenly distribute blocks of workstations and position endpoints around each for parallel agents operation.

In the size of S_{eval}, the NCA-generated manufacturing environment (Figure 13b) features standalone workstations surrounded by endpoints. We imitate this pattern to create the human-designed manufacturing environment of size S_{eval}, shown in Figure 14b. Similar to the human-designed environment of size S, we additionally let the ratio of different types of workstations to approximate $t_r : t_g : t_y = 2 : 5 : 10$ and the total number of workstations are similar.

Given these human-designed manufacturing environments have NCA-inspired designs, we anticipate them to outperform their counterparts in the warehouse domain, compared to the NCA-generated ones.

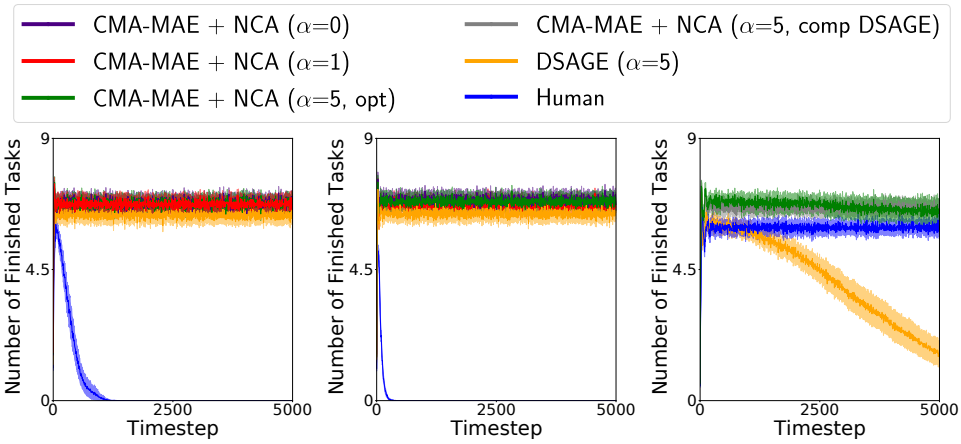
Maze. Figure 15 shows two example baseline environments using the method described Section 6. Compared to the NCA-generated maze environment of size S_{eval} shown in Figure 5b, both of them have no clear regularized patterns. Running the ACCEL agent in these two environments for 100 times results in success rates of 0% and 8%, respectively.

E Additional Results

In this section, we include the following additional results: (1) we compare the result archives with different α values to discuss the effect of α on patterns, (2) we show the number of finished tasks over time in the NCA-generated and baseline environments of size S in the warehouse and manufacturing domains, analyzing the occurrence of congestion in these environments, (3) we show the scalability of the trained NCA generators in environment sizes other than S and S_{eval}, (4) we compare our method with an additional baseline of tiling environments of size S to create those of size S_{eval}, (5) we show the QD-score and archive coverage of CMA-MAE + NCA comparing with MAP-Elites [34, 47] + NCA to demonstrate the advantage of CMA-MAE in terms of training a diverse collection of NCA generators, (6) we show the QD-score and archive coverage of CMA-MAE + NCA comparing with DSAGE with direct tile search [1, 51] to demonstrate the advantage of NCA in terms of generating environments with regularized patterns, and (7) we compare CMA-MAE with a derivative-free single-objective optimizer CMA-ES [20] to demonstrate that optimizing without diversifying measures can more easily lead to local optima.

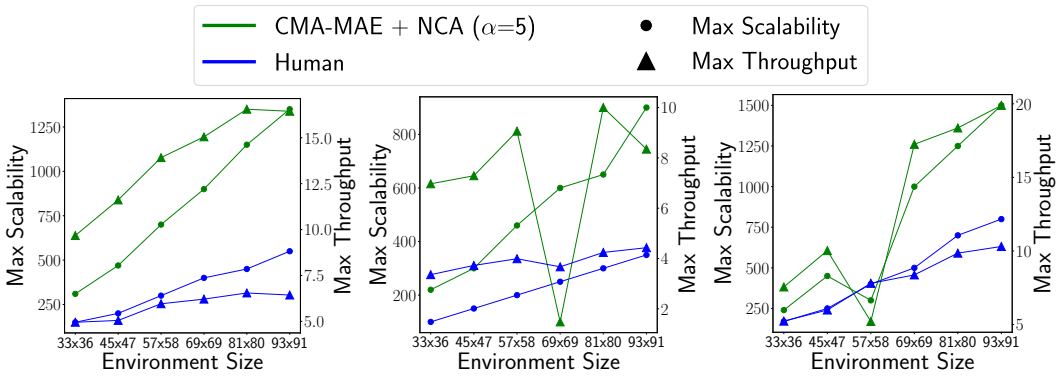
E.1 On the Effect of α on Patterns

Figure 16 shows the result archives of warehouse domains with different values of α. We observe that a larger α value is correlated with better archive coverage in the low environment entropy area in the archive. Since a lower value of environment entropy corresponds to clearer regularized patterns



(a) Warehouse (even) (b) Warehouse (uneven) (c) Manufacturing

Figure 17: Number of finished tasks per timestep with $N_a = 200$ agents. The solid lines are the average while the shaded area shows the 95% confidence interval.



(a) Warehouse (even) (b) Warehouse (uneven) (c) Manufacturing

Figure 18: Scalability of NCA-generated environments in the warehouse (even), warehouse (uneven), and manufacturing domains. To determine maximum scalability, we incrementally add agents and run 50 simulations, each with 5000 timesteps, for each count, monitoring average throughput. Once throughput drops, we identify the agent count corresponding to the peak mean throughput as the maximum scalability.

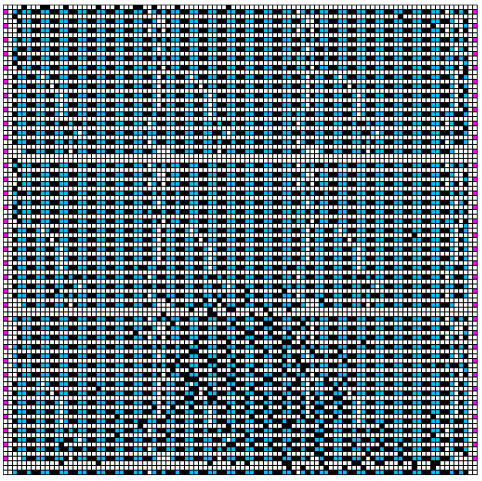
and thus a possibly more scalable NCA generator, a reasonably large α value can give users more freedom in choosing trained NCA generators from the result archive.

E.2 Number of Finished Tasks Over Time

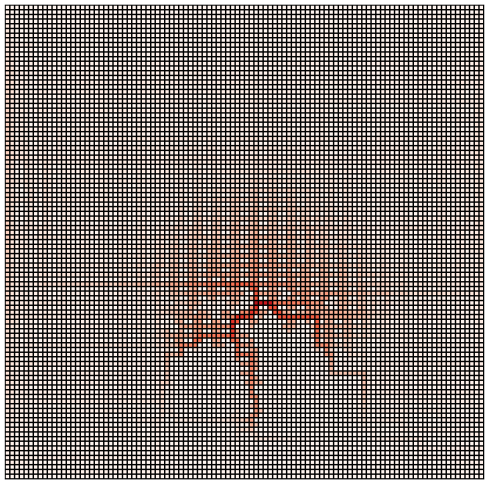
To understand the gap in the performance of the environments in the warehouse and manufacturing domains, we run 100 simulations with N_a agents in all of the environments of size S and plot the number of finished tasks at each timestep. We do not stop the simulation even if the agents encounter congestion. Figure 17 shows the number of finished tasks over 5,000 timesteps. All NCA-generated environments maintain a stable number of finished tasks throughout the simulation.

E.3 Scalability in More Environment Sizes

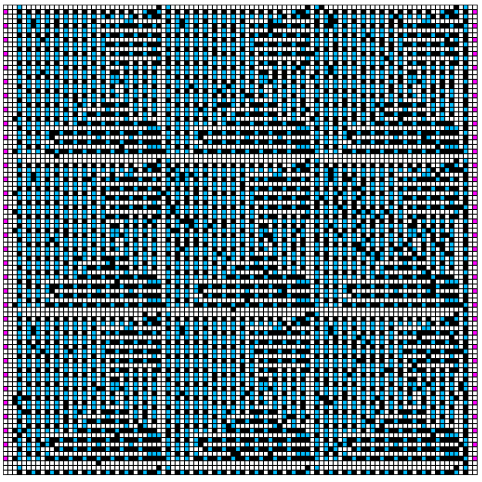
To further demonstrate the scalability of our method, we use trained NCA generators from Section 6 to generate progressively larger environments and run simulations. Figure 18 shows the result. The y-axis illustrates two metrics: maximum mean throughput over 50 simulations (right) and the maximum scalability, defined as the agent count at this maximum (left).



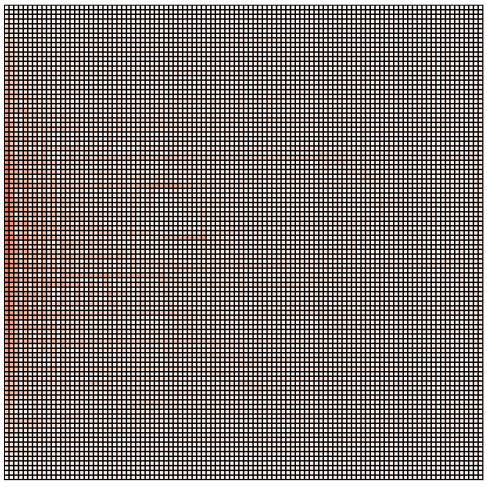
(a) Warehouse (even) tiling environment



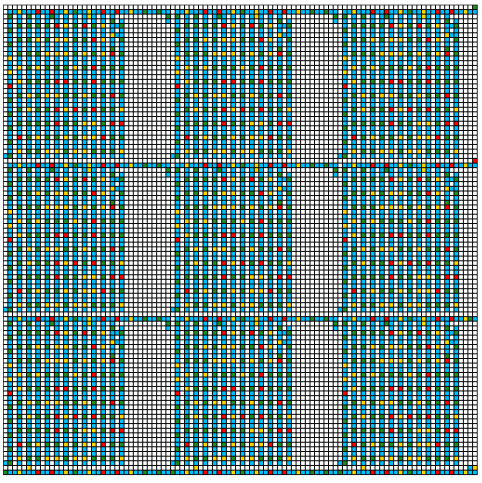
(b) Warehouse (even) tile-usage map



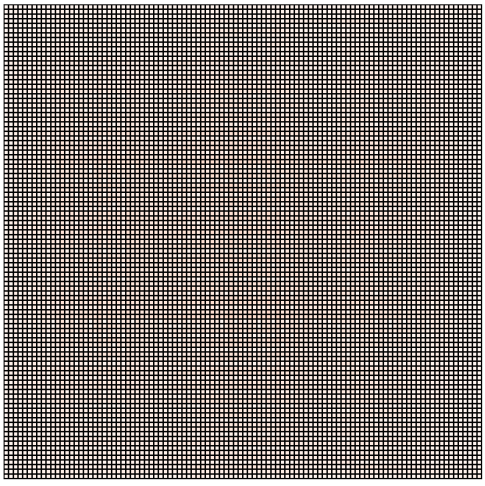
(c) Warehouse (uneven) tiling environment



(d) Warehouse (uneven) tile-usage map



(e) Manufacturing tiling environment



(f) Manufacturing tile-usage map

Figure 19: Baseline environments as well as tile-usage maps of size S_{eval} obtained by tiling smaller environments of size S.

Domain	Algorithm	QD-score	Archive Coverage
Maze w/o env entropy	**CMA-MAE + NCA**	**22,299.00 ± 612.46**	**0.54 ± 0.01**
	DSAGE	16,446.60 ± 42.27	0.40 ± 0.00
	MAP-Elites + NCA	12,446.80 ± 2,207.08	0.30 ± 0.05
Maze w/ env entropy	**CMA-MAE + NCA**	**12,468.20 ± 342.09**	**0.77 ± 0.02**
	DSAGE	1,444.20 ± 52.60	0.09 ± 0.00
	MAP-Elites + NCA	9,288.80 ± 590.13	0.57 ± 0.04
Warehouse (even)	**CMA-MAE + NCA**	**20,366.07 ± 653.46**	**0.33 ± 0.01**
	DSAGE	914.60 ± 33.09	0.02 ± 0.00
	MAP-Elites + NCA	6,893.49 ± 1,291.68	0.15 ± 0.03
Warehouse (uneven)	**CMA-MAE + NCA**	**19,985.23 ± 493.98**	**0.33 ± 0.01**
	DSAGE	926.69 ± 23.42	0.02 ± 0.00
	MAP-Elites + NCA	5,906.86 ± 1,027.29	0.14 ± 0.02
Manufacturing	**CMA-MAE + NCA**	**6,449.56 ± 1,106.83**	**0.18 ± 0.02**
	DSAGE	378.33 ± 4.69	0.02 ± 0.00
	MAP-Elites + NCA	2,398.99 ± 422.61	0.14 ± 0.02

Table 4: QD-score and Archive coverage. $x \pm y$ denotes an average value of x and a standard error of y over 5 runs. All algorithms use $\alpha = 5$. "Maze w/ env entropy" refers to using environment entropy and average agent path length as the measures in the maze domain, while "Maze w/o env entropy refers to using number of walls and average agent path length as the measures.

We see an increasing trend for both maximum scalability and maximum mean throughput as the environment size increases. The NCA-generated environments generally scale better than the human-designed ones.

We see two exceptions: the maximum mean throughput in the 69×69 warehouse (uneven) environment and both metrics in the 57×58 manufacture environment. We can attribute this to the interaction between the MILP and the specific environment generated by the NCA. Since MILP makes numerous changes to the generated environments in these domains, certain combinations of generated environments and MILP random seeds can lead to repaired layouts that create congestion. However, if we encounter such issues in practice, we can either leverage a different NCA generator from the archive or re-run the MILP repair with a different random seed.

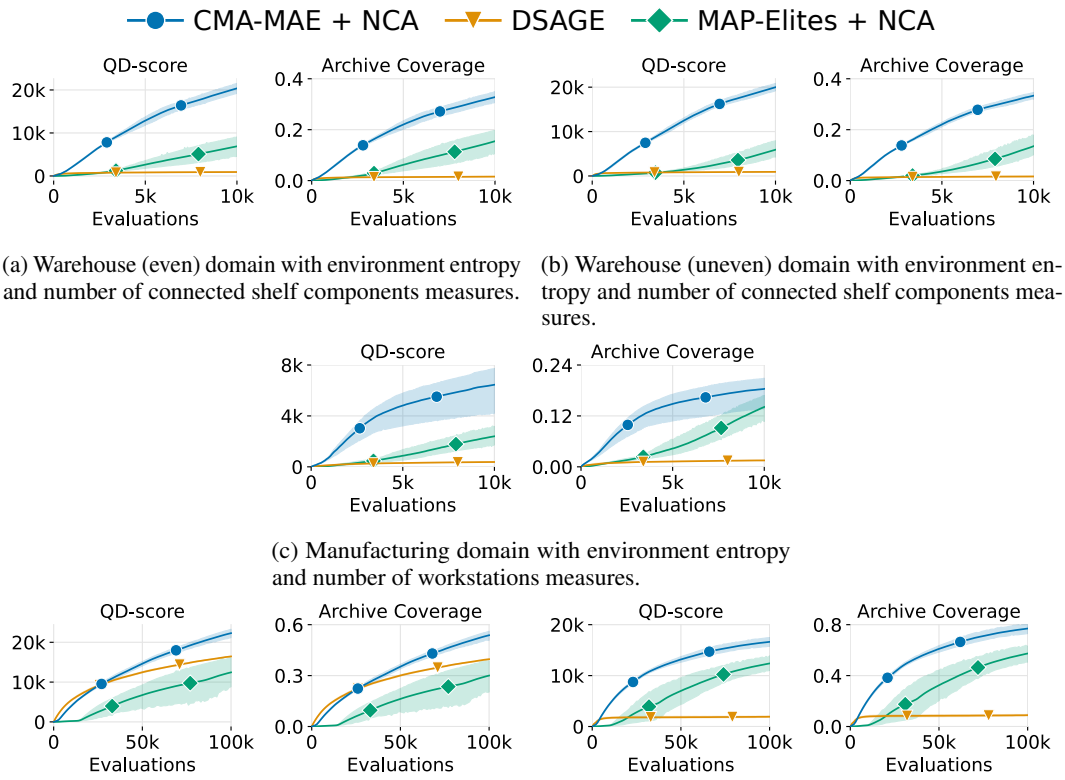
E.4 Tiling Environments of size S

Due to the similarity incurred in the NCA-generated environments of different sizes, one may argue that tiling small environments can create larger environments with competitive performance as the NCA-generated ones. To test this argument, we add a new baseline. We tile the environments of size S, namely Figure 10f (warehouse (even)), Figure 10i (warehouse (uneven)), and Figure 11d (manufacturing) in Appendix C.1, to create environments of size S_{eval} . We then use MILP to enforce constraints.

We run 50 simulations with N_{a_eval} agents specified in Table 1. The new baseline achieves a success rate of only 0% and 23% in the warehouse (even) and warehouse (uneven) domains, respectively. Figure 19 displays the tiled environments of size S_{eval} and their tile-usage maps, which show the usage frequency of each tile in the simulation. As shown in Figures 19b and 19d, the agents are congested, resulting in low success rates. In contrast, for the manufacturing domain, the baseline matches our method, with a success rate of 100% and an average throughput of 22.73. This is because the tiling of Figure 11d, creating Figure 19e, resembles the NCA-generated patterns in Figure 13b. Thus, the tiling baseline may be a good method for the manufacturing domain, yet it falls short in the warehouse domains.

E.5 QD Score and Archive Coverage

Figure 20 shows the QD-score and the archive coverage over the number of evaluations during training. Table 4 shows the corresponding numerical results. Figure 21 then shows the result archive of one of the runs.



(a) Warehouse (even) domain with environment entropy and number of connected shelf components measures.

(b) Warehouse (uneven) domain with environment entropy and number of connected shelf components measures.

(c) Manufacturing domain with environment entropy and number of workstations measures.

(d) Maze domain with number of walls and average agent path length measures.

(e) Maze domain with environment entropy and average agent path length measures.

Figure 20: QD-score and archive coverage of CMA-MAE + NCA, compared with MAP-Elites + NCA and DSAGE. The solid line shows the average and the shaded area shows the 95% confidence interval. All algorithms use $\alpha = 5$.

We compare CMA-MAE with MAP-Elites [34] to show the benefit of using CMA-MAE to train NCA. We use the state-of-the-art Iso+LineDD mutation operator [47] on MAP-Elites with $\sigma_{line} = 0.2$ and $\sigma_{iso} = 0.01$. We also compare CMA-MAE + NCA with DSAGE to demonstrate the benefit of using NCA to generate environments with regularized patterns. In the maze domain, in addition to the combination of diversity measures discussed in Section 5 (number of walls and average agent path length), we run experiments with the environment entropy measure, paired with average agent path length, to demonstrate the effect of the environment entropy measure on the QD-score and archive coverage.

We observe that CMA-MAE achieves the best QD-score and the best archive coverage in all domains, generating solutions with the best quality and diversity. We also observe from the result archives that CMA-MAE covers the largest number of cells. Notably, in all domains, DSAGE fails to diversify environment entropy, covering only the high entropy area of the archives and not exploring the low entropy region. This happens because we directly optimize environments instead of training NCA generators in DSAGE. Without NCA, DSAGE cannot generate candidate environments with low entropy (i.e., with regularized patterns). As a result, despite being high-quality, the DSAGE-optimized environments lack regularized patterns and cannot be scaled to arbitrary sizes.

E.6 Compare CMA-ES with CMA-MAE

Derivative-free single objective optimizers such as CMA-ES [20] that do not diversify a given set of measures can also be used to train the NCA generators. To demonstrate the effect of diversifying measures on the trained NCA generators, we run CMA-ES in the warehouse and manufacturing domains with $\alpha = 5$ and compare the trained NCA generator with those trained by CMA-MAE.

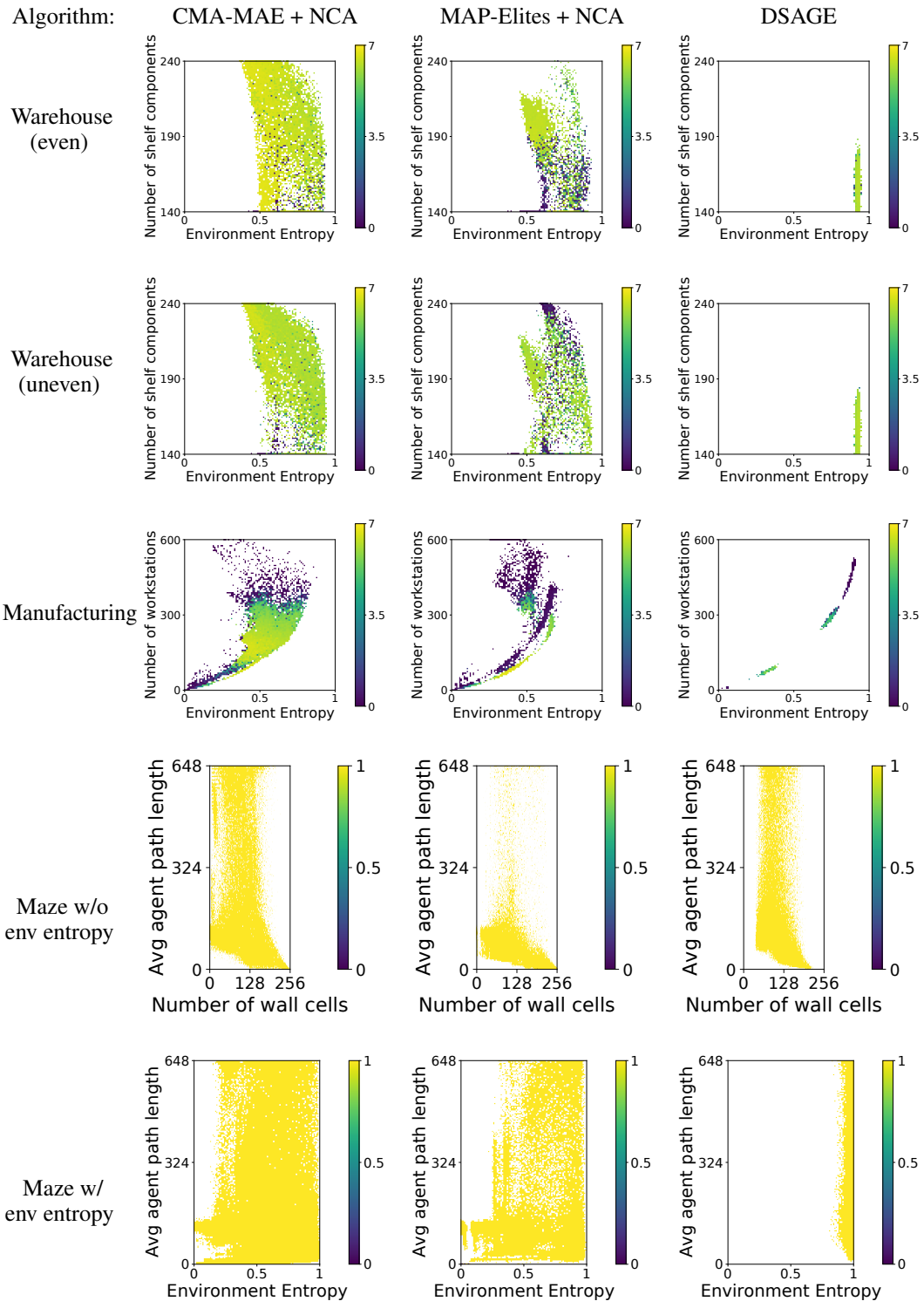
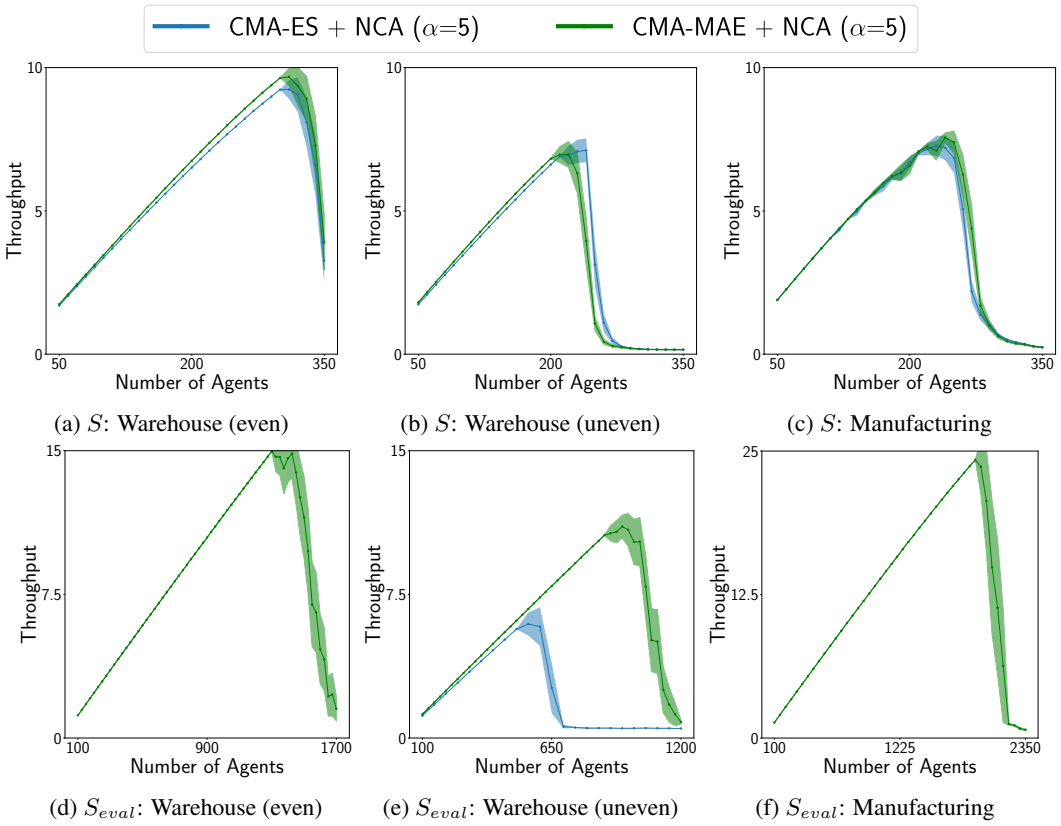


Figure 21: Example result archives of all domains. All algorithms use $\alpha = 5$.

Table 5 shows the numerical results, and Figure 22 shows the throughput over different numbers of agents.



(a) S: Warehouse (even) (b) S: Warehouse (uneven) (c) S: Manufacturing

(d) S_{eval}: Warehouse (even) (e) S_{eval}: Warehouse (uneven) (f) S_{eval}: Manufacturing

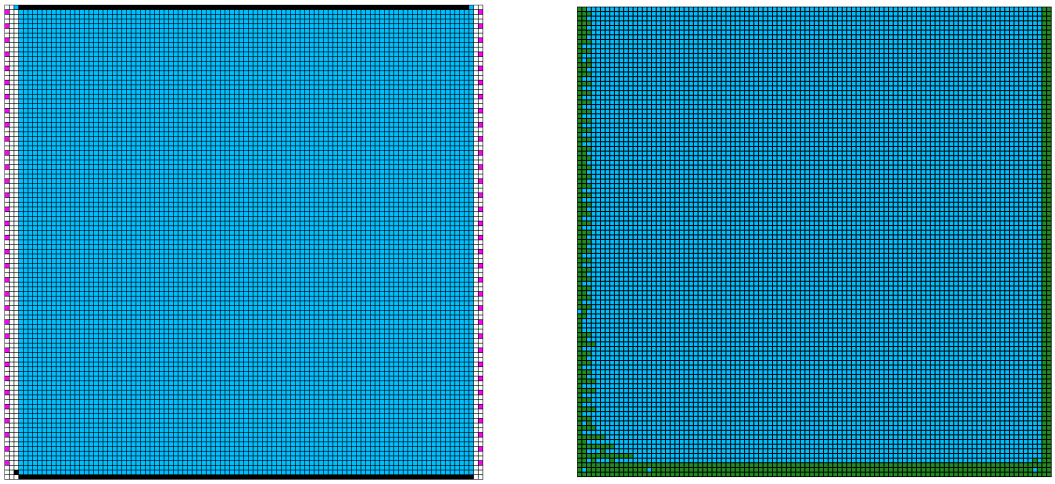
Figure 22: Throughput with an increasing number of agents in environments of sizes S and S_{eval} generated by CMA-ES + NCA and CMA-MAE + NCA. The NCA generators trained by CMA-ES fail to generate environments of size S_{eval} in the warehouse (even) and manufacturing domains because the unrepaired environments deviate too much from the domain-specific constraints. The solid lines are the average throughput while the shaded area shows the 95% confidence interval.

		Size S with N_a agents		Size S_{eval} with N_{a_eval} agents	
Domain	Algorithm	Success Rate	Throughput	Success Rate	Throughput
warehouse (even)	CMA-ES + NCA	**100%**	6.51 $\pm$ 0.00	N/A	N/A
	CMA-MAE + NCA	**100%**	**6.74 $\pm$ 0.00**	**90%**	**16.01 $\pm$ 0.00**
warehouse (uneven)	CMA-ES + NCA	**100%**	6.62 $\pm$ 0.00	0%	N/A
	CMA-MAE + NCA	**100%**	**6.82 $\pm$ 0.00**	**84%**	**12.03 $\pm$ 0.00**
Manufacturing	CMA-ES + NCA	**98%**	6.81 $\pm$ 0.01	N/A	N/A
	CMA-MAE + NCA	94%	**6.82 $\pm$ 0.00**	**100%**	**23.11 $\pm$ 0.01**

Table 5: Success rates and throughput of environments of sizes S and S_{eval} generated by CMA-ES + NCA and CMA-MAE + NCA. Both algorithms use $\alpha = 5$. We run 50 simulations for all environments except for the warehouse (uneven) environment of size S_{eval} generated by CMA-ES + NCA, for which we run 20 simulations. We measure the throughput of only successful simulations and report both average and standard error.

With an environment size of S, CMA-ES is competitive with CMA-MAE in all domains. In fact, CMA-ES ends up being slightly more scalable than CMA-MAE in the warehouse (uneven) domain with size S. However, CMA-MAE is significantly more scalable than CMA-ES with size S_{eval}. In particular, the MILP solver cannot find solutions for warehouse (even) and manufacturing domains of size S_{eval} in the given computational budget (introduced in Appendix C.4).

Figure 23 shows the unrepaired environments of size S_{eval} in the warehouse (even) and manufacturing domains for CMA-ES. Both environments have a large number of endpoints. As a result, they rely on the MILP solver to both satisfy the domain-specific constraints and generate patterns, which takes a significant amount of time because almost no constraints are satisfied in the unrepaired environments. Given the deviation of the unrepaired environments, CMA-ES fails to optimize the similarity score,



(a) Warehouse (even) (b) Manufacturing

Figure 23: Unrepaired NCA-generated environments of size S_{eval} in the warehouse (even) and manufacturing domains for CMA-ES + NCA ($\alpha = 5$).

converging to a local optima while optimizing only the throughput. In comparison, CMA-MAE is less prone to falling into the deceptive local optima because of the diversity measures.

F Societal Impact

We propose using QD algorithms to generate arbitrarily large environments that enhance throughput beyond state-of-the-art environment optimization methods and human-designed environments. This is achieved by optimizing a diverse collection of NCA generators. Our method can be applied to generate any multi-robot system as long as an agent-based simulator is available for evaluation. Large companies such as Amazon and Alibaba have deployed multi-robot systems in warehouses to transport packages or inventory pods. Therefore, one real world application of our method is optimizing the layout of the automated warehouses to improve throughput. Since our method is agnostic to the specific agent simulator and only requires metrics such as throughput post-simulation, we can plug-in different simulators and apply our environment generation algorithm. Improving the throughput of their warehouses can present a significant economic impact on the industry.

Our method may have negative impacts. While designing a real-world automated warehouse or manufacturing environment and deploying large-scale multi-robot systems in reality, we shall take into account other factors such as safety measures. In our method, however, we ignore all factors except for the throughput and scalability of the environment. Consequently, while our method can contribute to optimizing operational efficiency, caution should be exercised to ensure that other crucial parameters are not compromised in real-world applications.