Layout Design for Intelligent Warehouse by Evolution With Fitness Approximation

HAIFENG ZHANG [ID][1]**, ZILONG GUO** [ID][2]**, WEINAN ZHANG** [ID][3]**, HAN CAI**[4]**, CHRIS WANG**[5]**, YONG YU** [ID][3]**, WENXIN LI**[5]**, AND JUN WANG**[1]

[1]Department of Computer Science, University College London, London WC1E 6BT, U.K.
[2]Department of Computer Science, University of Southern California, Los Angeles, CA 90007, USA
[3]School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
[4]Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139, USA
[5]School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China

Corresponding author: Weinan Zhang (wnzhang@sjtu.edu.cn)

ABSTRACT With the rapid growth of the express industry, intelligent warehouses that employ autonomous robots for carrying parcels have been widely used to handle the vast express volume. For such warehouses, the warehouse layout design plays a key role in improving transportation efficiency. However, this work is still done by human experts, which is expensive and leads to suboptimal results. In this paper, we aim to automate the warehouse layout designing process. We propose a two-layer evolutionary algorithm to efficiently explore the warehouse layout space, where an auxiliary objective fitness approximation model is introduced to predict the outcome of the designed warehouse layout and a two-layer population structure is proposed to incorporate the approximation model into the ordinary evolution framework. Empirical experiments show that our method can efficiently design effective warehouse layouts that outperform both heuristic-designed and vanilla evolution-designed warehouse layouts.
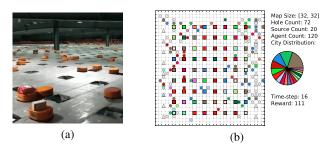
INDEX TERMS Evolutionary algorithm, intelligent warehouse, robots.

I. INTRODUCTION

The global express delivery industry has been a trillion market, serving people's daily life around the world. In 2017, the industry revenue is 248 billion USD [1] and in China, particularly, the annual gross express volume has surpassed 30 billion USD since 2016 [2]. During the recent two years, a new type of shipping warehouses, with intelligent robots sorting thousands of parcels per hour, emerged [3]. As shown in Figure 1a and 1b, autonomous robots carry parcels across the warehouse and unload the parcels into the target holes which connect to the vehicles heading to the target destinations. The layout of the warehouse, i.e. the matching of the holes and the target destinations, is usually designed by human experts. It can be challenging and also likely to be suboptimal, especially when the number of holes is large as shown in Figure 1b. Moreover, the demand of such warehouse layout design is not one-off, since the distribution of the parcel destinations is not fixed and the warehouse layout design should be adaptive to achieve the best performance.

In this paper, we present an evolution-based method for automatically designing warehouse layouts. To tackle the efficiency issue arising from the time-consuming evaluation of each designed warehouse layout, we consider training a neural network to predict outcomes of layouts without actually running agents in it, which is known as fitness approximation in the context of evolution [4]. We further propose a novel two-layer population structure to incorporate the prediction model into the evolution framework for improving efficiency, which can be categorized as multiple-deme parallel genetic algorithms [5]. Particularly, the higher layer consists of layouts that are actually evaluated and occupies a small fraction of the whole population while the lower layer contains layouts whose fitnesses are predicted by the learned model. Compared to existing methods for combining fitness approximation with evolution [6], [7], the proposed two-layer evolutionary algorithm explicitly manages evaluated individuals and predicted individuals separately in two sub-populations and trains the approximation model online using the samples evaluated by the original fitness function. As such, the proposed method incorporates fitness function approximation into the multiple-deme parallel genetic algorithm naturally. Moreover, within an evaluation of a designed

The associate editor coordinating the review of this manuscript and approving it for publication was Chun-Wei Tsai [ID].

(a)   (b)

FIGURE 1. (a) Real-world robotic warehouse for parcel sorting (screenshot from [3]). (b) Robotic warehouse environment. The triangles stand for the sources where parcels emerge. The circles stand for the robots carrying the parcels. The squares stand for the holes for the agents to put into the parcels. The squares are colored according to which destination the parcels coming into will go to. The agents repeatedly take a parcel with a color (destination) from a source to a hole with the same color. The objective is to maximize the total number of parcels processed by the agents in a fixed period.

warehouse layout, we can observe not only the final outcome but also additional agent trajectories that comprise hidden information about the causes of the outcome. To take advantage of such additional information to improve the quality of the prediction model, we construct an auxiliary objective, i.e. to predict the heatmap of the environment where each individual value is the total number of visits of a point.

Our experiments for designing warehouse layouts demonstrate improved efficiency and better performance compared to both manual design and vanilla evolution-based methods without fitness approximation. Such a two-layer evolution-based environment optimization framework is promising to be applied to various environment design tasks.

II. RELATED WORK
There are many real-world scenarios that can be regarded as environment design problems, ranging from game-level design with the desired level of difficulty [8], shopping space design for impulsing customer purchase and long stay [9] to traffic signal control for improving transportation efficiency [10]. In a recent work, [11] formulates these environment design problems using a reinforcement learning framework. In this paper, we focus on a new environment design scenario, i.e. warehouse layout design, emerging from the rapidly growing express industry.

Traditional warehouse design problems can be categorized to three levels, strategic level, tactical level, and operational level [12]. At the strategic level, long-term decisions are considered, including the size of a warehouse [13] and the selection of component systems [14]. At the tactical level, medium-term decisions are made, such as the layout of a conventional warehouse [15], [16]. At the operational level, detailed control policies are studied, e.g. batching [17] and storage policies [18]. The problem discussed in this paper is about warehouse layout design, which is at the tactical level traditionally. However, in the era of big data, the layout of warehouse could be adaptive to the changes of the external environment. Specifically, the layout of the warehouse could be redesigned at intervals according to the changing

destination distribution of the parcels. Thus, this problem is better to be categorized as an operational level problem.

For solving this problem, we adopt evolutionary algorithms. As getting a guiding signal means evaluating the designed objective in the target task, which would result in an unacceptable computational resource requirement for scenarios where evaluation is expensive. To reduce the number of expensive evaluations on real data needed before a satisfying result can be obtained, some works propose to learn a model to predict the outcome of a designed objective without actually running on real data [19], [20]. A similar idea has been explored in the field of evolution and is known as fitness approximation [4]. Due to the inaccuracy of fitness approximation, it is essential to use the approximation model together with the original fitness function [21], [22]. To incorporate the fitness model into the simulation-based evolutionary algorithms, individual-based [23] and generation-based [22] methods are studied. Differently, our approach explicitly manages two sub-populations whose individuals are evaluated by the approximation model and the original fitness function respectively. Similar approaches are known as multiple-deme parallel genetic algorithms [5]. Our work can be classified as a multiple-deme parallel genetic algorithm with a two-layer sub-population topology to balance exploitation and exploration.

III. PROBLEM DEFINITION
In this section, we formulate the environment design problem and introduce a particular robotic warehouse environment. We fix the agent policy in the robotic warehouse environment and focus on the remaining task, assigning destinations to the holes, which can be viewed as an environment design problem.

A. ENVIRONMENT DESIGN
In many scenarios, there are n agents taking actions in a designable environment, such as cars running in a transportation system, consumers shopping in a mall, and so on. Denote the ith agent's policy as π_i and the environment is parametrized as $\mathcal{M}_\theta = \langle S, A, T_\theta, R_\theta, \lambda \rangle$, where $S, A, T_\theta, R_\theta, \lambda$ denote state space, action space, transition function, reward function and reward discount respectively. After the agents play in the environment in an episode, a joint trajectory $H = \langle s_1, a_1, s_2, a_2, ... \rangle$ is produced and a cumulative reward G_i is given to the ith agent, where s_t and a_t denote state and joint action respectively. Moreover, the objective of the environment designer is given as $O(H)$, whose function form can be defined specifically, and the designer intends to design an optimal environment to maximize the expectation of its objective

$$\theta^* = \arg\max_\theta \mathbb{E}[O(H)|\mathcal{M}_\theta; \pi_{1...n}]. \quad (1)$$

Note that the randomness of H is derived from the possible randomness of π_i when selecting actions.

B. ROBOTIC WAREHOUSE ENVIRONMENT

In this paper, we consider a robotic warehouse environment abstracted from a real-world express system as shown in Figure 1a, where there is a warehouse for sorting parcels from a mixed input stream to separate output streams according to their respective destinations. The sorting process is done by the robots carrying parcels from the input positions (sources) to the appropriate output positions (holes) in the plane warehouse as Figure 1b illustrates. Note that Figure 1b simplifies the real scenario in Figure 1a into a discrete form. The acceleration and deceleration of the agents are ignored in the discrete form. In order to maximize the efficiency of sorting, we should set the robots' cooperative pathfinding algorithm and assign the destinations to the holes. In this task, the agents share a common reward G and the environment also takes G as its design objective, i.e. $O(H) = G$. We set π_ϕ as a joint policy model for the agents. As such, the problem is formulated as

$$\theta^*, \phi^* = \arg\max_{\theta, \phi} \mathbb{E}[G | \mathcal{M}_\theta, \pi_\phi]. \qquad (2)$$

For solving Eq. (2), we should firstly set a sound cooperative pathfinding algorithm π_{ϕ^*} for the robots. After, we focus on optimizing the environment parameter θ, i.e. optimizing the layout of the warehouse (the assignment of the destinations to the holes) via

$$\theta^* = \arg\max_\theta \mathbb{E}[G | \mathcal{M}_\theta; \pi_{\phi^*}]. \qquad (3)$$

Note that the demand for such environment layout design is not one-off. Since the external variables (such as the destination distribution of the parcels) may be changing, the best layout of the warehouse is changing accordingly. Thus, the layout of the warehouse should be redesigned at intervals, which gives a reason to find an efficient layout design approach.

C. DETAILED ENVIRONMENT DESCRIPTION

The warehouse is abstracted as a grid containing $h \times w$ cells. Among them, n_s cells are sources and n_h cells are holes, whose locations $l_s^{1..n_s}$, $l_h^{1..n_h}$ are given. There are n_r robots available to carrying parcels from sources to holes. Each cell is only for one robot to stand.

In each time-step, each robot is able to take a move to an adjacent cell. When an empty robot moves into a source, it loads a new parcel whose destination follows a distribution over n_d destinations (cities) with the proportions $p_1, p_2, \ldots, p_{n_d}$. On the other hand, when a loaded robot moves into a hole with the destination that is as the same as the loading parcel's, it unloads the parcel into that hole. That is to say, the rates of input and output flows are not restricted in our setting. Parcels are always sufficient when a robot moves into a source.

Our objective is to sort as many parcels as possible in a given time period T. We could achieve this objective by designing the layout of the warehouse, i.e. assigning the

proper destinations to the holes. Specifically, we should determine the parameter $\theta = \langle \theta_1, \theta_2, \ldots, \theta_{n_h} \rangle$ of the environment $\mathcal{M}_\theta$, where $\theta_i \in \{1..n_d\}$ for $i = 1..n_h$. Intuitively, the assignment of the destinations to the holes will affect the robots' paths and hence the efficiency of the whole warehouse.

The notations defined in this section are listed in Table 1.

TABLE 1. Notations and descriptions.

Notation	Description	Type
h	Height of warehouse	Input
w	Width of warehouse	Input
n_s	Number of source cells	Input
n_h	Number of hole cells	Input
$l_s^{1..n_s}$	Locations of source cells	Input
$l_h^{1..n_h}$	Locations of hole cells	Input
n_r	Number of robots	Input
n_d	Number of parcel destinations	Input
$p_{1..n_d}$	Proportions of parcel destinations	Input
T	Length of timestep	Input
$\theta_{1..n_h}$	Assignment of destinations to holes	Output

D. PROBLEM COMPLEXITY

For the problem defined above, the scale of the layout assignment space is $n_d^{n_h}$, where n_h denotes the number of the holes and n_d denotes the number of the parcel destinations. Since the robot pathfinding algorithm works like a black box to evaluate each layout assignment, it is hard to determine a global optimum without exploring the solution space completely. Thus, this optimization problem is an exponential time problem. Even for a small setting, such as $n_h = 20$, $n_d = 5$, the number of the assignments is as large as about 100 trillion, which is hard to be explored completely.

E. ROBOT PATHFINDING ALGORITHMS

In our problem, the robot pathfinding algorithm is fixed. As the robots are quite dense in the real-world warehouse, jam prevention is the key point. We considered two cooperative pathfinding algorithms with jam prevention design. The first one adopts WHCA* [24] as a planner, which searches the shortest path from an origin to a destination for each robot in turn and ensures non-collision. The second algorithm is a greedy one, which guides the robots by a lookup table in each position and reduces conflicts by setting one-way roads in the map as illustrated in Figure 2a. We studied these two algorithms and the results showed that the greedy one has a significant advantage on time complexity and a minor disadvantage on performance. Due to the large simulation demand for testing environment parameter, we selected the time-saving greedy algorithm as the agent policy in our experiments. However, the proposed warehouse layout design solution can work with other robot pathfinding algorithms as well.

IV. SOLUTION

In this section, we first introduce an evolution framework for automatically designing warehouse layout and then present the auxiliary objective fitness approximation and the two-layer population structure for improving the efficiency.
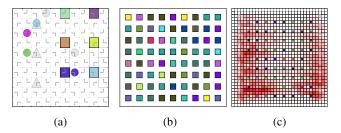
FIGURE 2. (a) An illustration of one-way roads: i) the odd-row cells allow moving right and forbid moving left, while the even-row cells allow moving left and forbid moving right; ii) the odd-column cells allow moving down and forbid moving up, while the even-column cells allow moving up and forbid moving down. The left-down cell is in Row 1 and Column 1. (b) A layout sample as an individual in the evolutionary algorithm. (c) An example of the heatmap.

A. EVOLUTION WITH ROBOT POLICY SIMULATION

Under the evolution framework, we maintain a population containing n warehouse layout individuals, i.e. assignments of the destinations to the holes (Figure 2b), and evolve the population for n_g generations. Within each generation, we perform crossover, mutation, and selection in the order:

- In the **crossover** phase, we randomly select c pairs of samples. For each pair of samples, we splice their holes from two matrices to two lines respectively. Then, we randomly select a common breakpoint for both lines and cross the two lines just like chromosomal crossover. Finally, we generate two square matrices by reshaping the two lines.
- In the **mutation** phase, we randomly select m_1 samples generated in the crossover phase. For each sample, we randomly select m_2 holes and randomly permute their destinations.
- In the **selection** phase, we evaluate the generated samples in the crossover and mutation phases by robot policy simulations, then merge the original and the generated samples. The best n ones are selected for the next generation.

B. TWO-LAYER EVOLUTIONARY ALGORITHM WITH FITNESS APPROXIMATION

In this section, we propose a novel evolutionary algorithm that trains an auxiliary objective fitness function to evaluate a large population for providing promising individuals to a small population evaluated by simulations.

1) AUXILIARY OBJECTIVE FITNESS APPROXIMATION

In practice, the simulation of robots performing in the environment is time-consuming. A promising way to reducing the simulation time is to use an approximation function to compute fitness:

$$f_G(\theta) = \hat{G} \approx \mathbb{E}[G|\mathcal{M}_\theta; \pi_{\phi^*}], \tag{4}$$

where f_G is the fitness approximation function, θ is a sample of environment parameter and $\hat{G}$ is the predicted fitness of θ, whose learning target is the expectation of the reward G.
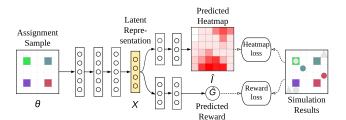
Moreover, since a simulation generates a trajectory H in addition to the reward G, we consider utilizing H to help training fitness function f_G. Although G is the exact objective for fitness function to learn, we may extract additional information $I(H)$ from H that helps training the fitness function, under the assumption that G and I are correlated. We set an auxiliary training objective and use a neural network to capture this:

$$f(\theta) = \langle f_I(f_X(\theta)), f_G(f_X(\theta)) \rangle = \langle \hat{I}, \hat{G} \rangle \tag{5}$$
$$\approx \langle \mathbb{E}[I(H)|\mathcal{M}_\theta, \pi_{\phi^*}], \mathbb{E}[G|\mathcal{M}_\theta, \pi_{\phi^*}] \rangle,$$

where f is a neural network consisting of three sub-networks: f_X is the bottom network that captures the common features and outputs X; f_I and f_G are the two separate networks on the top of X that predict $\hat{I}$ and $\hat{G}$ respectively.

In the robotic warehouse layout design problem, θ represents the assignment of the destinations to the holes and H represents the movements of the robots. Furthermore, we define I as the heatmap of the movements as Figure 2c shows. Intuitively, the distribution of busy areas should be correlated with the efficiency of sorting and the reward. The process of learning the fitness function in the warehouse layout problem is illustrated in Figure 3.

FIGURE 3. An illustration of the process of evaluating an assignment sample θ. First, the latent representation X is learned via shared deep layers. Then based on X, separated layers are built to predict heatmap $\hat{I}$ and reward $\hat{G}$ respectively. Two loss functions are calculated based on the difference between the prediction and the simulated results.

Since obtaining simulation samples is time-consuming, we train the fitness model online. Specifically, the fitness model is trained with the samples simulated along the process of the evolutionary algorithm. There is no pre-training in our approach.

2) TWO-LAYER POPULATION

The fitness model provides a less accurate but more speedy evaluation than the simulation. This property indicates that the simulation is better to find the local optimum exactly and the fitness model is better to explore the global space speedily. For the standard simulation-based evolution, mutation rate is usually set small enough to ensure convergence within an acceptable time, thus the search space is relatively local. Therefore, we consider incorporating the fitness model into the standard simulation-based evolution as an additional part of exploring the global space.

Specifically, we maintain two sub-populations. The first one is of the same size as the population set in the standard

Algorithm 1 Two-Layer Evolutionary Algorithm With Fitness Approximation (A Literal Expalnation of Figure 4)

Require: noble population N, civilian population C, untrained fitness model f, empty simulation sample set S

1: **for** each generation **do**
2:      generate N_1 from N by crossover and mutation;
3:      generate C_1 from C by crossover and mutation;
4:      rank $C \cup C_1$ by f to generate top population C_2, middle population C_3 and bottom population C_4;
5:      evaluate N_1 and C_2 by simulation and add the results to S;
6:      rank $N \cup N_1 \cup C_2$ by the simulation score to generate top population N_2 and bottom population N_3;
7:      generate random population R and discard C_4;
8:      pass N_2 to the next generation as N;
9:      pass $N_3 \cup C_3 \cup R$ to the next generation as C;
10:      update f using S.
11: **end for**

---

simulation-based evolution. Also, the individuals in the first sub-population are evaluated by simulations. The second sub-population is multiple times larger than the first one and the samples in it are evaluated by the fitness model. We view the second sub-population as a candidate population whose top individuals have a chance of joining the first sub-population. On the other hand, the bottom individuals in the first sub-population may be moved to the second sub-population. We name the first-layer sub-population noble and the second civilian. Noble population and civilian population evolve separately while keeping a channel for migration.

In detail, the two-layer population evolves as Figure 4 and Algorithm 1 show. In general, N and C maintain individuals evaluated by the simulation and the fitness model respectively. In each generation, migration takes place. Specifically, C_2 from the civilian layer go up to the noble layer and N_3 from the noble layer go down to the civilian layer. In addition, the civilian layer discards the worst population C_4 and absorbs a randomly generated population R.

There are 9 parameters related to the proposed two-layer evolutionary algorithm. They are noble population number $|N|$, civilian population number $|C|$, crossover rate c_N, c_C, mutation rate m_N, m_C, $|C_2|$ for the number of civilian
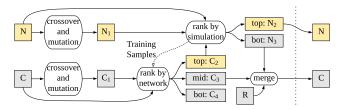

FIGURE 4. The process of the two-layer population evolutionary algorithm in a single generation. The yellow and grey squares stand for the populations who have been (or will be) evaluated by simulation and the fitness model respectively.

individuals migrate to the noble layer, $|R|$ for the number of the randomly generated individuals, and n_u for the number of model updates in each generation. Other variables can be determined by these parameters. In each generation, $|N_1| + |C_2|$ simulations, n_u model updates and $|C_1|$ model predictions are performed. Since the time cost of training the network and use it to predict is negligible compared to the simulations (see Table 4), the time complexity of the two-layer evolutionary algorithm for n_g generations is $O(n_g(|N_1| + |C_2|))$.

V. EXPERIMENT

We set up a virtual intelligent warehouse environment based on real-world settings and test our proposed approach comparing to the baselines.[1]

A. EXPERIMENT SETTINGS

Environment: We test our proposed approach in 20×20 maps. The positions of the sources and holes are set as real-world scenarios. The detailed parameters are given in Table 2. The destination distributions are set according to long-tail functions to reflect reality. In our experiments, the reward is defined as the sum of parcel loading times and unloading times (roughly two times as the number of parcels processed).

TABLE 2. Environment parameter settings.

h	w	n_s	n_h	n_r	n_d	T	$p_{1..n_d}$
20	20	12	20	60	5	1000	0.367, 0.267, 0.2, 0.133, 0.033

Robots: As introduced, we adopt a greedy algorithm as the cooperative pathfinding algorithm for the robots. Firstly, we set one-way roads in the map as Figure 2a shows to avoid opposite-directional conflicts, while right-angled conflicts are avoided by setting priority. On the one-way roads, the robots decide moves by a look-up table containing $h \times w \times (n_s + n_h)$ records, each of which indicates the first step towards a particular source or hole from a particular cell.

Baselines: We test 5 baselines to compare with our proposed two-layer evolutionary algorithm (**TLEA**). **Random**: The holes are assigned with random destinations uniformly. **Heuristic**: Destinations select holes in turns according to their proportions. For example, if 10% parcels are going to destination A, then A select 10% of the holes. This process starts from the destination with the most proportion. Each destination greedily selects each hole that minimizes the sum of the average distance from the sources to the selected holes. **Simu**: The evolutionary algorithm with simulations as introduced in the Solution section. **SimuInd**: An implementation of the individual-based evolution control algorithm [23]. This approach maintains a single large population for evolution

---

[1]Our experiment is repeatable and the source code is provided on the GitHub: shorturl.at/gmBI9.
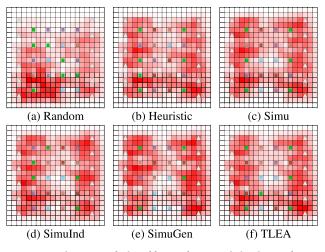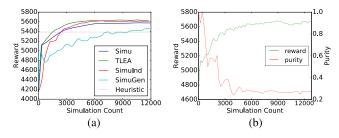
FIGURE 5. Environments designed by Random, Heuristic, Simu and TLEA.

(a) Random (b) Heuristic (c) Simu (d) SimuInd (e) SimuGen (f) TLEA

FIGURE 6. (a) Learning curves averaged over 10 runs. The Y-axis is the reward received by the best individual in each population. (b) Impact of the civilian population for a particular run. Initially, the purity of each individual in the noble population is set to be 1 and each civilian is set to be 0. During the evolution, each child's purity is the mean of its parents' purity.

whose individuals are evaluated by the fitness model. In each generation, the best individuals evaluated by the fitness model are evaluated by the simulation once again. The fitness model is trained online with the samples produced by the simulations. **SimuGen**: An implementation of the generation-based evolution control algorithm [22]. This approach also maintains a single large population as SimuInd. The difference is that SimuGen uses the simulations intensively in a generation and uses the fitness model in the next several generations.

Hyper-Parameters: To ensure fairness, for Simu, SimuInd, SimuGen, and TLEA, the number of generations is set as 60 and the number of simulations in each generation is set as 200. The model update and prediction times are also fixed as 5000 and 10000 respectively for SimuInd, SimuGen and TLEA. The population of Simu is 100; in each generation, 200 individuals are generated by crossover; 50 of them are mutated. For SimuInd and SimuGen, the populations are 5000; 10000 are generated by crossovers in each generation; 2500 of them are mutated. For the TLEA, $|N|, |C|, c_N, c_C, m_N, m_C, |C_2|, |R|, n_u$ are set to be 100, 5000, 1, 1, 0.25, 0.25, 50, 2500, 5000 respectively.

Fitness Model: Our network is composed of three sub-networks f_X, f_I, f_G. The output of f_X is used for the input of f_I and f_G. f_X has two fully connected layers whose output is a vector that can be reshaped to match the size of the map. Then, a 2D transposed convolution layer follows. f_I has one transposed convolution layer to generate the heat map. And f_G contains three fully connected layers to predict the reward. All the layers except the output layers have a ReLU activation function. The loss functions for the two outputs are set to be MSE. The first two fully connected layers have 128, 400 units respectively. The first 2D transposed convolution layer has 16 filters. And the second one has one filter. The three fully connected layers for reward prediction have 256, 128 and 1 unit respectively.

Hardware: We use two computers with an Intel core i7-4790k and an Intel core i7-6900k respectively. The one with 4790k also has an extra Nvidia Titan X GPU.

B. RESULTS
We perform the baselines and TLEA. The results are shown in Table 3. We find Heuristic is fairly high compared to Random but is inferior to evolutionary algorithms. Moreover, TLEA outperforms all the baselines.

TABLE 3. Performance of Random, Heuristic, Simu, SimuInd, SimuGen and TLEA. The algorithms are repeatedly performed for 10 runs. The reward samples pass the Shapiro-Wilk test to be normal. T-tests are performed for TLEA against Simu, SimuInd and SimuGen. The statistical results show that the superiority of TLEA is significant.

	Reward	T Score	P-Value
Random	4757	-	-
Heuristic	5386	-	-
Simu	5572	5.8778	7×10^{-6}
SimuInd	5605	2.7708	6.3×10^{-4}
SimuGen	5499	5.8782	7×10^{-6}
TLEA	**5646**	-	-

Figure 5 shows the layouts designed by the baselines and TLEA with the heatmaps. We can see that the tracks of the robots running in the maps of TLEA are better balanced, indicating that there are fewer traffic jams.

Figure 6a shows the learning curves. Since SimuInd and SimuGen mix the individuals evaluated by the simulation and the fitness model, their current best individuals may be the over-estimated ones by the inaccurate fitness model, which may lead to discarding the real best individuals. TLEA solves this problem by separating the two populations and ensure that the real best individual is always kept in the noble population.

In addition, TLEA and Simu are more stable than SimuInd and SimuGen, because the temporary best individual may be evaluated by the fitness model in SimuInd and SimuGen, which may be corrected by the simulation in later generations. The slight fluctuations of Simu and TLEA are caused by the variance of the simulations, which results in that the best samples can be over-estimated (which is much slighter than the fitness model) and would be averaged by extra simulations in later generations.

C. DISCUSSIONS
Time Cost: The time costs of the tested algorithms are listed in Table 4. It shows that the time cost proportion of the fitness

TABLE 4. Time cost comparison. The average time costs for simulation, model update and model predicting are 2.62s, 2.42ms and 1.06ms respectively. The number of generations is 60 for all the algorithms.

	Simu	SimuInd	SimuGen	TLEA
Simulation	$12k$	$12k$	$12k$	$12k$
Model Update	0	$300k$	$300k$	$300k$
Model Predicting	0	$600k$	$600k$	$600k$
Time	$8.73h$	$9.11h$	$9.11h$	$9.11h$

model is less than 5%. In our experiment, we just ignore the time difference between Simu and other algorithms.

Effectiveness of Heatmap: We evaluate randomly generated samples by the simulations and use them to train the fitness functions with and without heatmaps as auxiliary objective. We compare MSE and Pearson Correlation of them in Table 5, which shows that heatmap provides significant improvement to the fitness function.

TABLE 5. Comparison of fitness functions with and without heatmap.

Sample Number	MSE		Pearson Correlation	
	w/o	Heatmap	w/o	Heatmap
5000	4.36	2.89(-33.72%)	0.277	0.519(+87.36%)
10000	2.75	1.59(-42.18%)	0.405	0.687(+69.63%)
20000	1.67	0.69(-58.68%)	0.766	0.908(+18.54%)

Simulation Allocation: Since simulations are scarce resources when running the evolutionary algorithm, the allocation of simulations between the noble layer and the civilian layer is important. Moreover, it also determines the migration rate between the two layers. We test different $\frac{|N_1|}{|N_1|+|C_2|}$, the ratio of simulations allocated to the noble layer, and find that 0.75 is a proper setting (see Table 6), which means three fourths simulations are allocated to ensure the accuracy of the noble layer and one fourth simulations are allocated to give chances to the civilian layer.

TABLE 6. Simulation allocation analysis.

Noble Proportion	0.25	0.5	0.75	1
Reward	5629	5634	**5646**	5581

Impact of Civilian Population: We are interested in how much contribution has the civilian population made to the evolution of the noble population. We calculate a number named purity that measures how much the evolved noble population inherits from the initial noble population. As Figure 6b shows, the purity of the noble population declines rapidly along with the increasing of the reward (fitness). Finally, the civilian population contributes more than 70 percent to the noble population.

Limitation of the Algorithm: In general, the proposed TLEA can be used in environment design problems that have massive environment parameters. Although the algorithm is able to work with simple hyper-parameters, the optimal hyper-parameters rely on problem domain knowledge and need fine-tuning.

VI. CONCLUSION

In this paper, we study the problem of automatic warehouse layout design. The proposed two-layer evolutionary algorithm takes advantage of a fitness approximation model, augmented with an auxiliary objective of predicting the heatmap. Our approach enhances the exploration of the evolutionary algorithm with the help of the fitness model. The experiments demonstrate the superiority of our approach over the heuristic and the traditional evolution-based methods. For future work, we would apply the proposed two-layer evolutionary algorithm to other environment design scenarios, such as shopping mall design, game design, and traffic light control.
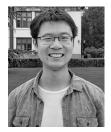
REFERENCES

[1] IBISWorld. (2018). *Global Courier & Delivery Services Industry—Market Research Report.* [Online]. Available: goo.gl/h6fdWq

[2] W. Fan, M. Xu, X. Dong, and H. Wei, "Considerable environmental impact of the rapid development of China's express delivery industry," *Resour., Conservation Recycling*, vol. 126, pp. 174–176, Nov. 2017.

[3] C. People's Daily. (2017). *Robots Sorting System Helps Chinese Company Finish at Least 200,000 Packages a Day in the Warehouse.* [Online]. Available: goo.gl/hLbYhV

[4] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Comput.*, vol. 9, no. 1, pp. 3–12, Jan. 2005.

[5] E. Cantú-Paz, "A survey of parallel genetic algorithms," *Calculateurs Paralleles, Reseaux Syst. Repartis*, vol. 10, no. 2, pp. 141–171, 1998.

[6] E. D. de Jong, D. Thierens, and R. A. Watson, "Hierarchical genetic algorithms," in *Proc. Int. Conf. Parallel Problem Solving Nature*. Berlin, Germany: Springer, 2004, pp. 232–241.

[7] Y.-S. Hong, H. Lee, and M.-J. Tahk, "Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks," *Eng. Optim.*, vol. 35, no. 1, pp. 91–102, 2003.

[8] J. Togelius, G. N. Yannakakis, K. O. Stanley, and C. Browne, "Search-based procedural content generation: A taxonomy and survey," *IEEE Trans. Comput. Intell. AI in Games*, vol. 3, no. 3, pp. 172–186, Sep. 2011.

[9] A. Penn, "The complexity of the elementary interface: Shopping space," in *Proc. 5th Int. Space Syntax Symp.*, vol. 1, 2005, pp. 25–42.

[10] H. Ceylan and M. G. H. Bell, "Traffic signal timing optimisation based on genetic algorithm approach, including drivers' routing," *Transp. Res. B, Methodol.*, vol. 38, no. 4, pp. 329–342, 2004.

[11] H. Zhang, J. Wang, Z. Zhou, W. Zhang, Y. Wen, Y. Yu, and W. Li, "Learning to design games: Strategic environments in reinforcement learning," in *Proc. IJCAI*, 2018, pp. 3068–3074.

[12] B. Rouwenhorst, B. Reuter, V. Stockrahm, G.-J. van Houtum, R. J. Mantel, and W. H. M. Zijm, "Warehouse design and control: Framework and literature review," *Eur. J. Oper. Res.*, vol. 122, no. 3, pp. 515–533, 2000.

[13] Y. Roll, M. J. Rosenblatt, and D. Kadosh, "Determining the size of a warehouse container," *Int. J. Prod. Res.*, vol. 27, no. 10, pp. 1693–1704, 1989.

[14] A. Keserla and B. A. Peters, "Analysis of dual-shuttle automated storage/retrieval systems," *J. Manuf. Syst.*, vol. 13, no. 6, pp. 424–434, 1994.

[15] Y. Bassan, Y. Roll, and M. J. Rosenblatt, "Internal layout design of a warehouse," *AIIE Trans.*, vol. 12, no. 4, pp. 317–322, 1980.

[16] J. R. Berry, "Elements of warehouse layout," *Int. J. Prod. Res.*, vol. 7, no. 2, pp. 105–121, 1968.

[17] E. A. Elsayed and R. G. Stern, "Computerized algorithms for order processing in automated warehousing systems," *Int. J. Prod. Res.*, vol. 21, no. 4, pp. 579–586, 1983.

[18] M. Goetschalckx and H. D. Ratldff, "Optimal lane depths for single and multiple products in block stacking storage systems," *IIE Trans.*, vol. 23, no. 3, pp. 245–258, 1991.

[19] B. Baker, O. Gupta, R. Raskar, and N. Naik, "Accelerating neural architecture search using performance prediction," 2018, *arXiv:1705.10823*. [Online]. Available: https://arxiv.org/abs/1705.10823

[20] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," 2017, *arXiv:1712.00559*. [Online]. Available: https://arxiv.org/abs/1712.00559

[21] D. E. Grierson and W. H. Pak, "Optimal sizing, geometrical and topological design using a genetic algorithm," *Struct. Optim.*, vol. 6, no. 3, pp. 151–159, 1993.

[22] A. Ratle, "Accelerating the convergence of evolutionary algorithms by fitness landscape approximation," in *Proc. Int. Conf. Parallel Problem Solving Nature*. Berlin, Germany: Springer, 1998, pp. 87–96.

[23] L. Bull, "On model-based evolutionary computation," *Soft Comput.*, vol. 3, no. 2, pp. 76–82, 1999.

[24] D. Silver, "Cooperative pathfinding," in *Proc. AIIDE*, vol. 1, 2005, pp. 117–122.

CHRIS WANG received the B.S. degree from the University of Alberta, Canada. He is currently pursuing the master's degree with Peking University. His research interests include artificial intelligence, machine learning, and robotics.
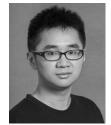
HAIFENG ZHANG received the B.S. degree in computer science from Peking University, the second bachelor's degree in economics from Peking University, in 2012, and the Ph.D. degree from Peking University, in 2018, under the supervision of Prof. W. Li. He is currently a Research Fellow with University College London, under the supervision of Prof. J. Wang. His research areas include reinforcement learning, game AI, game theory, and computational advertising.

YONG YU is currently a Professor with the Department of Computer Science, Shanghai Jiao Tong University. His research interests include information systems, web search, data mining, and machine learning. He has published over 200 articles and a dozen of other related conferences (e.g., NIPS, ICML, SIGIR, and ISWC) in these fields. He served as PC Member for several conferences, including WWW and RecSys.

ZILONG GUO received the bachelor's degree from Shanghai Jiao Tong University. He is currently pursuing the master's degree with the University of Southern California. His research interests include artificial intelligence, machine learning, and reinforcement learning.
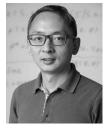
WENXIN LI received the B.S., M.S., and Ph.D. degrees from the Department of Computer Science and Technology, Peking University, in 1990, 1993, and 2001, respectively, and the second Ph.D. degree from the Department of Computing, The Hong Kong Polytechnic University, in 2004. She has been the Associate Dean of the School of EECS, since 2010. She is currently a Professor with the Department of Computer Science and Technology, School of EECS. Her research interests include biometrics, image processing, pattern recognition, content-based image retrieval, and education.

WEINAN ZHANG is currently an Assistant Professor with the Department of Computer Science, Shanghai Jiao Tong University. He has published 50 research articles on conferences and journals, including KDD, SIGIR, AAAI, WWW, WSDM, ICDM, JMLR, and IPM. His research interests include machine learning and data mining, particularly, deep learning and reinforcement learning techniques for real-world data mining scenarios.

JUN WANG is currently a Chair Professor of computer science with University College London and the Founding Director of the M.Sc. Web science and big data analytics. He is also a Co-Founder and the Chief Scientist with MediaGamma Ltd., a UCL start-up company focusing on AI for intelligent audience decision making. His main research interests are in the areas of AI and intelligent systems, including (multiagent) reinforcement learning, deep generative models, and their diverse applications on information retrieval, recommender systems and personalization, data mining, smart cities, bot planning, and computational advertising.

HAN CAI received the B.S. and M.S. degrees from the School of Electronics Information and Electrical Engineering, Shanghai Jiao Tong University, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree with the Massachusetts Institute of Technology. He has over ten publications, including AAAI, NIPS, and ICLR. His research interests are mainly in the areas of machine learning, reinforcement learning, and AutoML.

● ● ●